Understanding Enthymemes in Argument Maps: Bridging Argument Mining and Logic-based Argumentation

Jonathan Ben-Naim, Victor David, Anthony Hunter

Understanding Enthymemes in Argument Maps: Bridging Argument Mining and Logic-based Argumentation

Jonathan Ben-Naim[1] Victor David[2] Anthony Hunter[3]
[1]Université Paul Sabatier, CNRS, IRIT - Toulouse, France
[2]Université Côte d'Azur, Inria, I3S - Sophia Antipolis, France
[3]University College London - London, UK

Abstract

Argument mining is natural language processing technology aimed at identifying arguments in text. Furthermore, the approach is being developed to identify the premises and claims of those arguments, and to identify the relationships between arguments including support and attack relationships. In this paper, we assume that an argument map contains the premises and claims of arguments, and support and attack relationships between them, that have been identified by argument mining. So from a piece of text, we assume an argument map is obtained automatically by natural language processing. However, to understand and to automatically analyse that argument map, it would be desirable to instantiate that argument map with logical arguments. Once we have the logical representation of the arguments in an argument map, we can use automated reasoning to analyze the argumentation (e.g. check consistency of premises, check validity of claims, and check the labelling on each arc corresponds with the logical arguments). We address this need by using classical logic for representing the explicit information in the text, and using default logic for representing the implicit information in the text. In order to investigate our proposal, we consider some specific options for instantiation.

1 Introduction

Argument mining aims to identify the parts of text that represent arguments (premises and/or claims), and the support or attack relationships between those arguments [LR19]. Using a range of natural language processing (NLP) technology, with increasing emphasis on large language models, they can be trained to identify arguments and relationships between them with increasing accuracy. The resulting information can then be represented by an argument graph or argument map, where each node contains the text representing the premises and claim of the argument, and each arc denotes a relationship (such as positive/support, or negative/attack) between arguments, as illustrated in Figures 1 and 2. In this paper, we assume the difference between an argument graph and argument map is the latter distinguishes between the premises and the claim of each argument.

Whilst argument graphs and maps provide a useful visualizable summary of argumentation in text, there is then a lack of automated methods for analysing or reasoning with the information in the argument map. To address this shortcoming, it would be desirable to translate the text into logical arguments, and then use automated reasoning to analyze the arguments (e.g. check consistency of premises, check validity of claims, and check the labelling on each arc corresponds with the logical arguments assigned to the abstract arguments in the map). Unfortunately, there is a lack of a formal representational framework for bridging argument maps and the underlying logical arguments. So this gives us the first problem that we tackle in this paper:

(**Problem 1**) *How can we represent the explicit information in an argument map, and then represent that explicit information in a logical rendition of the arguments?*

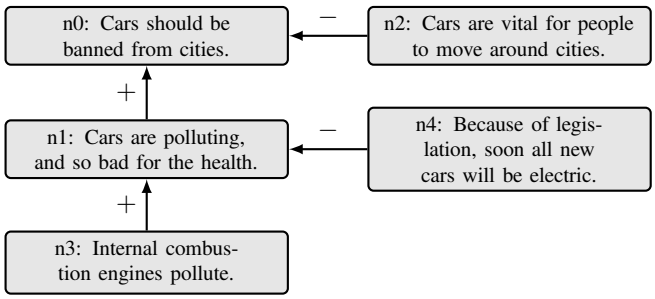


Figure 1: Example of an argument graph where each node in $\{n0, n1, n2, n3, n4\}$ represents an argument, and the text exposition is given after the colon. The $+$ (resp. $-$) label on an arc denotes a support (resp. attack) relationship. According to the text, $n0$ appears to be a claim without premises, though $n1$ appears to provide support for what might be implicit premises of $n0$. Also $n1$ appears to have premises for its own claim. So what are the implicit premises of $n0$ and how do they entail the claim of $n0$? And what does the claim of $n1$ entail for supporting the implicit premise of $n0$? We can also ask about how the claim of $n2$ attacks $n0$. Is $n2$ attacking the claim of $a0$ or is it attacking the implicit premises of $n0$? We can ask the same kind of questions for the remaining arguments and arcs here.

There are a number of frameworks for modelling argumentation in logic (e.g. [BH05, GH11, MP14, Ton14, AD18, CLS20]). They incorporate a formal representation of individual arguments, where the premises imply the claim, and techniques for comparing conflicting arguments [ABG$^+$17]. However, real arguments (i.e. arguments presented by humans) usually do not have enough explicitly presented premises for the entailment of the claim. This is because there is some common or commonsense knowledge[1] that can be assumed by a proponent of an argument and the recipient of it [Wal01].

Whilst human agents constantly need to understand enthymemes, whether in everyday or professional life, there is a lack of adequate algorithmic methods for supporting or automating this process. The coding/decoding can be modelled as abduction [Hun07, BH12, HMR14], and there has been some consideration of how this can be undertaken within a dialogue [BH08b, dS11b, dS11a, XHMB20, PMB22, LGG23]. However, these proposals do not provide a systematic framework for translating argument maps into logic, and concomitantly, addressing the problem of identifying the missing premises and/or claim, and discerning the relationships between them. We illustrate this problem with example of an argument graph in Figure 1, and in the argument map in Figure 2 where we flag the premises and claims that are completely implicit with the "Null" value. This brings us to the second problem we tackle in this paper:

> (**Problem 2**) *How can we represent the implicit information pertaining to the arguments in an argument map within a logical rendition of the arguments?*

To address these two questions in this paper, we start from the assumption that each premise, and each claim, is a string of text, such as a phrase, a sentence, or a paragraph. If the premise or claim is implicit, then we use the Null value. We then assume a function that can translate each piece of text into a formula of classical logic, where the formula is an atom, a propositional formula, or a first-order formula, depending on the nature and granularity of translation. We use classical logic because it has a clear syntax and semantics for representing and reasoning with information. Then, for each argument we use default logic (which we introduce in the next paragraph) for connecting each claim with its premise, and each argument with the arguments it supports or attacks.

Default logic was developed as a formalism for commonsense reasoning. It extends classical logic with default rules [Rei80]. Default rules are a generalization of natural deduction rules. They provide a clear

[1]Commonsense knowledge is knowledge normally known by everyone, i.e. universally known (as opposed to local knowledge). For instance, it is commonsense knowledge that if drop an egg on the floor, it will probably break, or if you tell your friends a funny joke, they will probably laugh. For a review of commonsense knowledge representation and reasoning see [Dav17]. In contrast, common knowledge is knowledge that is known by a subset of people? For example, if two old friends are talking and one of them refers a past event that they had experienced together, without explicitly recalling it, they may both have implicit but common knowledge of the event which would be a context known only to relatively few people (non-universal knowledge).

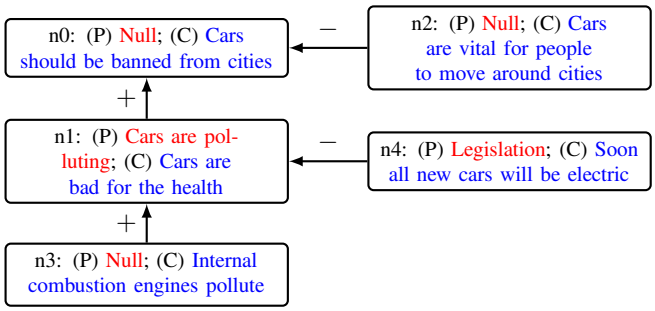


Figure 2: Continuing Figure 1, an argument map where each node in $\{n0, n1, n2, n3, n4\}$ is an argument. After the colon, the text for the premise follows (P) and text for the claim follows (C). The $+$ (resp. $-$) on an arc denotes a support (resp. attack) relationship.

representation of how default inferences can be obtained, and of how this reasoning can be attacked. Whilst there are previous proposals for using default logic in logic-based argumentation (e.g. [Pra93, SM08, Hun18]), none of these proposals provide a comprehensive coverage of the diversity of attack and support relations that can be captured, and moreover, none of these proposals consider how implicit information as arising in enthymemes can be handled, nor how argument maps can be instantiated by logic-based arguments.

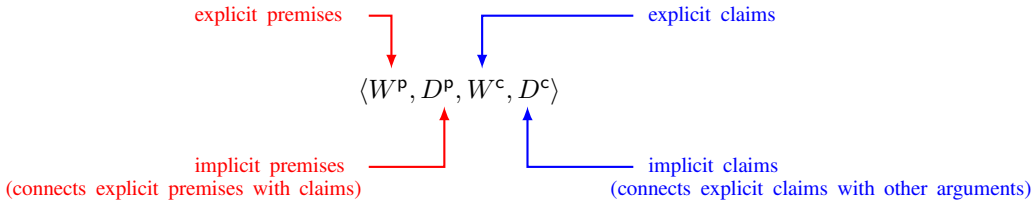


Figure 3: Structure of a default argument

We introduce the notion of a default argument as a tuple $\langle W^{\mathsf{p}}, D^{\mathsf{p}}, W^{\mathsf{c}}, D^{\mathsf{c}} \rangle$ where W^{p} is a set of classical formulae that represents the explicit premises of the argument, D^{p} is a set of default rules that denote the implicit premises of the argument, W^{c} is a set of classical formulae that represent the explicit claims of the argument, and D^{c} is a set of default rules that denote the implicit claim of the argument.

For each node in the argument map, we identify an appropriate logical argument to instantiate the node. This results in an instantiated argument map which can be analysed to check the validity of arguments, and of the labels on the arcs. Furthermore, different instantiations for the same argument map can be compared (e.g. to determine whether one is finer-grained than another), and different maps can be compared in terms of their instantiated version (e.g. to determine whether one is equivalent to another in some sense). This brings us to the third problem we tackle in this paper:

> (**Problem 3**) *How do we analyze and compare argument maps that are instantiated with default arguments?*

In the rest of this paper, we review default logic, and then use it to define the notion of a default argument, and to define different notions of support and attack between arguments. We then define how we bridge argument maps and logical argumentation by translating the sentences in argument maps into classical formulae, and then instantiating the argument maps with logical arguments. Once we have presented our novel framework, we compare it with the related literature. We conclude with a discussion of our proposal and of future work. We provide a summary overview of our pipeline in Figure 4.

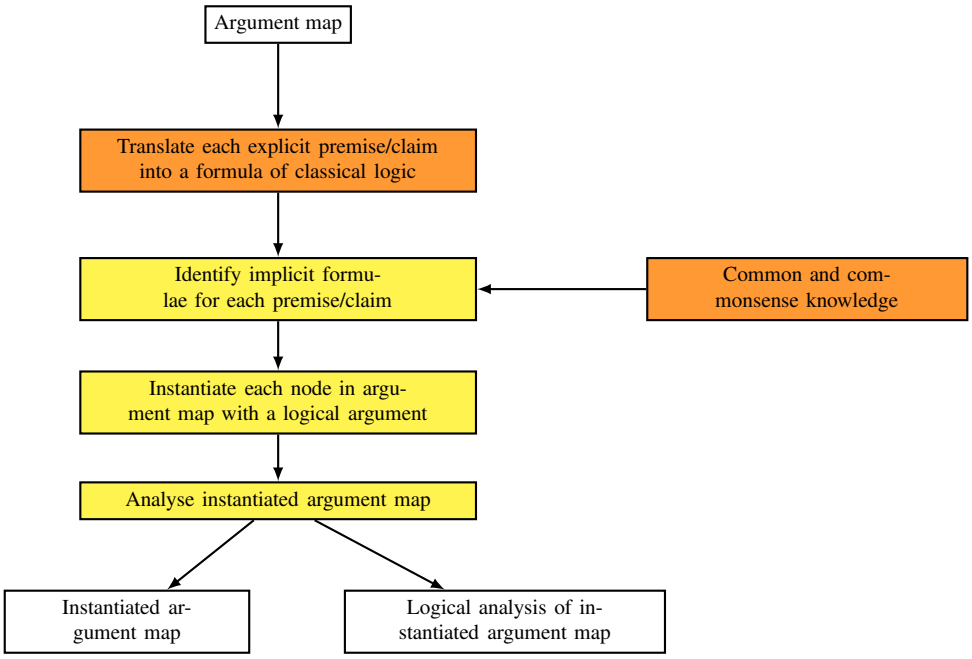


Figure 4: Overview of our pipeline for understanding enthymemes. The input is the white box at the top, and the outputs are the the white boxes at the bottom. The focus of this paper is on the knowledge representation and reasoning aspects of the yellow boxes (i.e. how do we represent the explicit and implicit aspects of a logical argument, how do we represent the instantiation of an argument map with logical arguments, and what kinds of logical analyse can we undertake with the instantiation of an argument map). We leave the underlying mechanisms for these yellow boxes to future work (i.e. algorithms for identifying implicit formulae for each premise and claim, algorithms for instantiating each node in argument map with a logical argument, and algorithms for automated analysis of instantiated argument maps). We also leave the orange boxes to future work (i.e. the automated natural language understanding of premises and claims of natural language arguments, the acquisition of commonsense knowledge, and knowledge representation and reasoning with commonsense knowledge).

2 Translation of explicit premise and claim into logic

We assume the usual propositional and predicate (first-order) languages for classical logic. Let $\mathcal{A}$ be the set of ground atoms (i.e. propositional atoms and positive ground literals). Let $\mathcal{P}$ be the set of propositional formulae composed from atoms $\mathcal{A}$ and the logical connectives $\wedge$, $\vee$, $\neg$. Let $\mathcal{F}$ be the set of first-order formulae composed from a set of function symbols, predicate symbols, the logical connectives $\wedge$, $\vee$, $\neg$, and the quantifiers $\forall$ and $\exists$. Since we are often indifferent as to whether we use the propositional or first-order language, we use $\mathcal{L}$ to denote a language that can be either a propositional or first-order language. In the examples, we will use teletype font for the logical formulae.

2.1 Argument maps

We assume an argument map is obtained as output from argument mining of a text (e.g. a discussion document). Each node denotes an argument (often with some/all of the premises and/or claim being implicit). Often the text in a node appear as a statement rather than a complete argument since either the premises or claim are implicit. We represent an argument map as a tuple as follows.

Definition 1. *Let $\mathcal{T}$ be a set of text strings. An* **argument map** *is a tuple (N, P, C, L) where N is a set of nodes; $P : N \rightarrow \mathcal{T} \cup \{\text{Null}\}$ is a text labelling function for premises; $C : N \rightarrow \mathcal{T} \cup \{\text{Null}\}$ is a text labelling function for claims; And $L : N \times N \rightarrow \{+, -, \square\}$ is an arc labelling function, where $+$*

N	P	C
n0	Null	Cars should be banned from cities.
n1	Cars are polluting	Cars are bad for the health.
n2	Null	Cars are vital for people to move around cities.
n3	Null	Internal combustion engines pollute.
n4	Legislation	Soon all new cars will be electric.

Figure 5: An argument map (N, P, C, L) where the graph (N, A) and the labelling function L is given in Figure 2, and the P and C functions for the arguments are in the table.

represents an attack relationship, $-$ represents a support relationship, and $\square$ represent no relationship holds from the first node to the second node.

Each premise text and each claim text is a phrase, or sentence, or paragraph of text. It communicates the explicit information required for the premise or claim of each argument. So the starting point for a formalization of argumentation should be the representation of these strings of text. There is a long history of graphical representation of arguments and counterarguments where text associated with each argument and counterargument. An argument map is an example of such a representation and we can consider the output of argument mining as being an argument map.

2.2 Logical translation

Given an argument map, we want to represent each of the premise and the claim by a set of formulae of classical logic. The following definition of translation function is a key step in bridging the output of argument mining (i.e. the text-based annotation of argument maps) and a logic-based representation of the argument map.

Definition 2. *Let $\mathcal{T}$ be a set of text strings. A* **logical translation** *is a function $T : \mathcal{T} \cup \{\text{Null}\} \to \wp(\mathcal{L})$ s.t.*[2] *$T(\text{Null}) = \emptyset$.*

If there is no text string (i.e. Null), then the translation is $\emptyset$ (i.e. tautology) which represents no useful information.

Example 1. *Containing Example 2, the following logical translation T is listed for the text string for each premise and claim. In this example, each text string is assigned an atom in the logical language.*

$n0$	$T(\text{Null}) = \emptyset$	$T(\text{Cars should be banned from cities}) = \{\texttt{s1}\}$
$n1$	$T(\text{Cars are polluting}) = \{\texttt{s3}\}$	$T(\text{Cars are bad for the health}) = \{\texttt{s0}\}$
$n2$	$T(\text{Null}) = \emptyset$	$T(\text{Cars are vital for people to move around cities}) = \{\texttt{s2}\}$
$n3$	$T(\text{Null}) = \emptyset$	$T(\text{Internal combustion engines pollute}) = \{\texttt{s4}\}$
$n4$	$T(\text{Legislation}) = \{\texttt{s5}\}$	$T(\text{Soon all new cars will be electric}) = \{\texttt{s6}\}$

The following are some types of translation, where the first is the most abstract and the simplest to implement, and the third is the least abstract of the three, and it is the most difficult to implement.

Atomic translation. The logical translation is a set of propositional atoms. So each text string is assigned an atom (as illustrated in Example 1). This allows for different ways that a claim is expressed in text being assigned to the same atom (as illustrated in Example 2). This can be implemented by a text string classifier where each string is assigned an atom that denotes its class. Assignments can group similar text items to the same atom as in intent classification [GTF+23] or key point analysis [BEK+21]. Also, in an argumentative chatbot system, sentence embeddings have been used to classify a user input as to being one of around 1500 arguments in an argument graph in order to identify a counterargument to be given by the chatbot [CH20].

[2] $\wp$ stands for power-set.

Propositional translation. The logical translation is the set of propositional formulae that can be formed from a set of propositional atoms (as illustrated in Example 3). A translation function can be implemented using natural language processing that can identify the underlying Boolean connectives in a sentence (e.g.[WvEH16, LLG+22]).

First-order translation. The logical translation is the set of first-order formulae that can be formed from a set of predicate and function symbols (as illustrated in Example 4). A translation function could then be implemented using natural language processing such as an AMR parser [DCS17, DZF+22], neural network [SAK20], or reinforcement learning [LLG+22].

Example 2. *Consider the text for two arguments "we should stop quantitative easing because government printing of money can cause prices to rise" and 'we should stop quantitative easing because quantitative easing can cause inflation". Assume that by using argument mining applied to the first argument, the text for the premise is "government printing of money can cause prices to rise" and applied to the second argument, the text for the premise is "quantitative easing can cause inflation". These premises may be regarded as making the same point. This can be implemented by a text string classifier where each string is assigned an atom that denotes its class.*

$$T(\textit{"government printing of money can cause prices to rise"}) = \mathtt{a_{13}}$$
$$T(\textit{"quantitative easing can cause inflation"}) = \mathtt{a_{13}}$$

Example 3. *Consider the text for an argument "quantitative easing causes inflation, which is highly undesirable, and so we should stop quantitative easing". Assume that by using argument mining applied to this argument, the text for the premise is "quantitative easing causes inflation, which is highly undesirable", and the text for the claim is "we should stop quantitative easing". The following is a propositional translation of the premise and claim.*

$$T(\textit{"quantitative easing causes inflation, which is highly undesirable"}) = (\mathtt{quantitative\_easing} \rightarrow \mathtt{inflation}) \wedge \neg \mathtt{inflation}$$

$$T(\textit{"we should stop quantitative easing"}) = \neg \mathtt{quantitative\_easing}$$

Example 4. *Consider the text for an argument "we should stop quantitative easing because whenever a country uses long-term quantitative easing, there is high inflation". Assume that by using argument mining applied to this argument, the text for the premise is "whenever a country uses long-term quantitative easing, there is high inflation". The following is a first-order translation of the premise.*

$$T(\textit{"whenever a country uses long-term quantitative easing, there is high inflation"}) = \forall \mathtt{x}\ \mathtt{country}(\mathtt{x}) \wedge \mathtt{quantitative\_easing}(\mathtt{x}, \mathtt{long\_term}) \rightarrow \mathtt{inflation}(\mathtt{x}, \mathtt{high})$$

The size of the codomain of the logical translation relative to the domain determines the granularity of the translation. For example, if the domain is large, but we only have relatively few formulae in the codomain, then we have a coarse-grained translation.

3 Review of default logic

We use $\alpha, \beta, \gamma, \delta, \phi, \psi, \ldots$ for arbitrary formulae and $\Delta, \Gamma, \ldots$ for arbitrary sets of classical formulae. We let $\vdash$ denote the classical consequence relation, and write $\Delta \vdash \bot$ to denote that Δ is inconsistent. $\mathsf{Atoms}(\Delta)$ gives the atoms appearing in the formulae in Δ. Let Cn be the consequence closure function (i.e. $\mathsf{Cn}(\Delta) = \{\phi \mid \Delta \vdash \phi\}$). For $\phi, \psi \in \mathcal{L}$, $\phi \equiv \psi$ denotes that ϕ and ψ are equivalent (i.e. $\{\phi\} \vdash \psi$ and $\{\psi\} \vdash \phi$). For $\Delta, \Gamma \subseteq \mathcal{L}$, $\Delta \equiv \Gamma$ denotes that Δ and Γ are equivalent (i.e. $\mathsf{Cn}(\Delta) = \mathsf{Cn}(\Gamma)$).

As a basis of representing and reasoning with default knowledge, default logic, proposed by Reiter [Rei80], is one of the best known and most widely studied formalisations of default reasoning. Furthermore, it offers a very expressive and lucid language.

In default logic, knowledge is represented as a set of propositional or first-order formulae and a set of default rules for representing default information. A **default rule** is of the following form, where α, β and γ are classical formulae.

$$\frac{\alpha : \beta}{\gamma}$$

For this, α is called the **pre-condition**, β is called the **justification**, and γ is called the **consequent**, of the default rule.

A **default theory** is a pair (D, W) where D is a set of default rules and W is a set of classical formulae. Default logic extends classical logic. Hence, all classical inferences from the classical information in a default theory are derivable (if there is an extension as defined below). The default theory then augments these classical inferences by default inferences derivable using the default rules: If α is inferred, and $\neg\beta$ cannot be inferred, then infer γ. The following is a definition for when a default theory has an extension.

Definition 3. *Let (D, W) be a default theory, where D is a set of default rules and W is a set of classical formulae. The operator* Pe *indicates what conclusions are to be associated with a given set E of formulae, where E is some set of classical formulae. For this, $\mathsf{Pe}(E)$ is the smallest set of classical formulae such that the following three conditions are satisfied.*

1. $W \subseteq \mathsf{Pe}(E)$,
2. $\mathsf{Pe}(E) = \mathsf{Cn}(\mathsf{Pe}(E))$,
3. *For each default in D, where α is the pre-condition, β is the justification, and γ is the consequent, the following holds: if $\alpha \in \mathsf{Pe}(E)$, and $\neg\beta \notin E$, then $\gamma \in \mathsf{Pe}(E)$.*

We refer to E as the **satisfaction set***, and $\mathsf{Pe}(E)$ the* **potential extension***. Furthermore, E is an* **extension** *of (D, W) iff $E = \mathsf{Pe}(E)$.*

We explain the above definition as follows: if E is an extension, then the first condition ensures that the set of classical formulae W is also in the extension, the second condition ensures the extension is closed under classical consequence, and the third condition ensures that for each default rule, if the pre-condition is in the extension, and the justification is consistent with the extension, then the consequent is in the extension. Minimality ensures that the extension is grounded (i.e. it avoids one or more default rules being self-supporting).

Example 5. *Let D be the following set of defaults. Note, each default in this example is given as a schema where the variables in the default rule are grounded before use to give a set of propositional default rules.*

$$\frac{\mathtt{bird(X)} : \neg\mathtt{penguin(X)} \wedge \mathtt{fly(X)}}{\mathtt{fly(X)}} \qquad \frac{\mathtt{penguin(X)} : \mathtt{bird(X)}}{\mathtt{bird(X)}} \qquad \frac{\mathtt{penguin(X)} : \neg\mathtt{fly(X)}}{\neg\mathtt{fly(X)}}$$

For (D, W), where W is $\{\mathtt{bird(Tweety)}\}$, we obtain one extension

$$\mathsf{Cn}(\{\mathtt{bird(Tweety)}, \mathtt{fly(Tweety)}\})$$

For (D, W), where W is $\{\mathtt{penguin(Tweety)}\}$, we obtain one extension

$$\mathsf{Cn}(\{\mathtt{penguin(Tweety)}, \mathtt{bird(Tweety)}, \neg\mathtt{fly(Tweety)}\})$$

Example 6. *Let D be the following set of defaults.*

$$\frac{\mathtt{bird(X)} : \mathtt{fly(X)}}{\mathtt{fly(X)}} \qquad \frac{\mathtt{penguin(X)} : \neg\mathtt{fly(X)}}{\neg\mathtt{fly(X)}}$$

For (D, W), where W is $\{\mathtt{bird(Tweety)}, \mathtt{penguin(Tweety)}\}$, we obtain two extensions

$$\mathsf{Cn}(\{\mathtt{bird(Tweety)}, \mathtt{fly(Tweety)}\}) \qquad \mathsf{Cn}(\{\mathtt{penguin(Tweety)}, \neg\mathtt{fly(Tweety)}\})$$

The notion of an extension in default logic is different to that of the notion of extension in abstract argumentation. In the former, an extension is a set of formulae obtained from a default theory, whereas in the latter, an extensions is an acceptable set of arguments from an argument graph. However, as shown by Bondarenko *et al.* [BDKT97], default logic can be modelled in assumption-based argumentation that is in turn based on abstract argumentation. But despite this underlying relationship between default logic and argumentation, there remains the need to address the questions raised in the introduction of this paper concerning the need for mechanisms for translation of argument maps into structured argumentation, and understand how implicit knowledge can be obtained and represented in order to understand enthymemes. This in turn calls for a more comprehensive study of how different kinds of argument, including enthymemes, and attack and suppport relationships between them, can be represented in a formalism based on default logic. We will explore these issues in the rest of this paper.

3.1 Types of default theory

There are various special cases of default rules that will be useful for our purposes. These include the following.

- **Precondition-free default rule.** This is a default rule of the form $\top : \beta/\gamma$ and so the consequent is obtained as long as the justification is satisfiable. For example, we might use the following default rule to capture the reasoning that anything that is unbroken is usable.

$$\top : \neg\mathtt{broken}(\mathtt{X})/\mathtt{useable}(\mathtt{X})$$

- **Justification-free default rule.** This is a default rule of the form $\alpha : \top/\gamma$ and so the consequent is obtained as long as the precondition holds, and so there is no block by the justification. This would mean that we assume that there are no exceptions. For example, we might use the following default rule to capture the reasoning that anything that is divisible by two is even.

$$\mathtt{divisablebytwo}(\mathtt{X}) : \top/\mathtt{even}(\mathtt{X})$$

- **Normal default rule.** This is a default rule of the form $\alpha : \beta/\beta$ and so the consequent is obtained when the precondition holds, and the consequent is satisfiable. For example, we might use the following default rule to capture the reasoning that if it is consistent to believe that someone has no brother, then we infer they have no brother.

$$\mathtt{person}(\mathtt{X}) : \neg\mathtt{hasBrother}(\mathtt{X})/\neg\mathtt{hasBrother}(\mathtt{X})$$

- **Semi-normal default rule.** This a default rule of the form $\alpha : \beta \wedge \gamma/\gamma$ and so the consequent is obtained when the precondition holds, and the justification which includes the consequent is satisfiable. For example, we might use the following default rule to capture that a bird flies if it is consistent to believe that it flies and that it is not a penguin (as we used in Example 5).

$$\frac{\mathtt{bird}(\mathtt{X}) : \neg\mathtt{penguin}(\mathtt{X}) \wedge \mathtt{fly}(\mathtt{X})}{\mathtt{fly}(\mathtt{X})}$$

For more coverage of default logic, see [Bes89, Bre91, BDK97]. There are algorithms for automated reasoning with default logic [BQQ83, RS94], and scalable implementations of default logic [TDJ22]. Also, a default theory can be translated into an answer set program (ASP) and an ASP solver used to automate the reasoning [CWZZ10].

3.2 Singular default theories

In the next section, we use default logic in our definition for default arguments. For this, the only restriction is that the default rules in the premises give a unique extension, and for this we introduce the following subsidiary definition.

Definition 4. *A default theory (D, W) is* **singular** *iff there is a unique extension of (D, W). When a default theory (D, W) is singular, let $\mathsf{Ex}(D, W)$ denote the extension.*

Example 7. *The default theory $(\{(\mathtt{a} : \mathtt{b}/\mathtt{b}), (\mathtt{b} \vee \mathtt{c} : \mathtt{d} \wedge \mathtt{f}/\mathtt{e})\}, \{\mathtt{a}\})$ is singular, and so there is a unique extension $\mathsf{Ex}(\{(\mathtt{a} : \mathtt{b}/\mathtt{b}), (\mathtt{b} \vee \mathtt{c} : \mathtt{d} \wedge \mathtt{f}/\mathtt{e})\}, \{\mathtt{a}\}) = \mathsf{Cn}(\{\mathtt{a}, \mathtt{b}, \mathtt{e}\})$. In contrast, the default theory $(\{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{b}), (\mathtt{a} : \neg\mathtt{b}/\neg\mathtt{b})\})$ is not singular, as there are two extensions $\mathsf{Cn}(\{\mathtt{a}, \mathtt{b}\})$ and $\mathsf{Cn}(\{\mathtt{a}, \neg\mathtt{b}\})$.*

The default theories $(\emptyset, W)$ and $(D, \{\bot\})$, for any D, are singular. For the former, $\mathsf{Ex}(\emptyset, W) = \mathsf{Cn}(W)$, and for the latter, $\mathsf{Ex}(D, \{\bot\}) = \mathsf{Cn}(\{\bot\})$. Also, if (D, W) is singular, then for all $D' \subseteq D$, then (D', W) is singular.

Proposition 1. *(Proposition 6.2.23 in [Bes89]) For a default theory (D, W), (D, W) is singular if W is consistent with $\{\beta \wedge \gamma \mid \alpha : \beta/\gamma \in D\}$.*

Proof. Let X be the smallest superset of W that is deductively closed and has the property that for any $\alpha : \beta/\gamma \in D$, if $\alpha \in X$, then $\gamma \in X$. So, X exists and is unique. It remains to prove that X is an extension of (D, W). Let $X = \mathsf{Cn}(W \cup Y)$ where Y is the set of all formulae introduced by means of the defaults in D. So $Y \subseteq \{\beta \wedge \gamma \mid \alpha : \beta/\gamma \in D\}$. Hence, $\{\beta \mid \alpha : \beta/\gamma \in D\}$ is consistent with $\mathsf{Cn}(W \cup Y)$. So for any $\alpha : \beta/\gamma \in$, $\neg\beta \notin X$. Also if $\alpha \in X$, then $\gamma \in X$. Therefore, we get the following applicability property: for any $\alpha : \beta/\gamma \in D$, if $\alpha \in X$, and $\neg\beta \notin X$, then $\gamma \in X$. Since X is the smallest set such that $W \subseteq X$, and $\mathsf{Cn}(X) = X$, and the applicability property holds, then X is an extension of (D, W). □

The converse of the above proposition does not necessarily hold. For instance, if W is inconsistent, then (D, W) is singular but W is not consistent with $\{\beta \wedge \gamma \mid \alpha : \beta/\gamma \in D\}$,

More importantly for our purposes is that any default theory with an extension E can be turned into a singular default theory by removing defaults without changing the extension E.

Proposition 2. *For a default theory (D, W), if E is an extension of (D, W), and (D, W) is not singular, then there is a default theory (D', W) s.t. $D' \subseteq D$ and E is an extension of (D', W), and (D', W) is singular.*

Proof. Either $\bot \in E$ or $\bot \notin E$. First assume $\bot \in E$. But this holds if and only if $W \vdash \bot$. Therefore let $D' = \emptyset$. Therefore (D', W) is singular. Now assume $\bot \notin E$. Let $X = \{\alpha : \beta/\gamma \mid \alpha \in E \text{ and } \neg\beta \notin E\}$. Let $F(X) = \{\gamma \mid \alpha : \beta/\gamma \in X\}$. So $E = \mathsf{Cn}(F(X) \cup W)$. Let D' be X. So E is the unique extension of (D', W). □

Example 8. *Let $D = \{\mathtt{a} : \mathtt{b}/\mathtt{b}, \mathtt{a} : \neg\mathtt{b}/\neg\mathtt{b}\}$ and $W = \{\mathtt{a}\}$. So there are two extensions from (D, W) which are $E_1 = \{\mathtt{a}, \mathtt{b}\}$ and $E_2 = \{\mathtt{a}, \neg\mathtt{b}\}$. The subtheory (D_1, W) where $D_1 = \{\mathtt{a} : \mathtt{b}/\mathtt{b}\}$ is singular with the extension being E_1, and the subtheory (D_2, W) where $D_2 = \{\mathtt{a} : \neg\mathtt{b}/\neg\mathtt{b}\}$ is singular with the extension being E_2.*

So if (D, W) is singular, and E is the extension of (D, W), then for every default in $\alpha : \beta/\gamma \in D$, the default contributes to the extension (i.e. $\gamma \in E$), or the default is not applied (i.e. $\alpha \notin E$), or the default can never be applied with W (i.e. $W \cup \{\beta\} \vdash \bot$), as illustrated in the following example.

Example 9. *Let $D = \{\mathtt{a} : \mathtt{p}/\mathtt{q}, \mathtt{b} : \neg\mathtt{a}/\mathtt{r}, \mathtt{c} : \mathtt{s}/\mathtt{s}\}$ and $W = \{\mathtt{a}, \mathtt{b}\}$. Therefore, (D, W) is singular with the extension being $\mathsf{Cn}(\{\mathtt{a}, \mathtt{b}, \mathtt{q}\})$ and where the first default (i.e. $\mathtt{a} : \mathtt{p}/\mathtt{q}$) is applied, the second default (i.e. $\mathtt{b} : \neg\mathtt{a}/\mathtt{r}$) is not applied, and the third default can never be applied with W (i.e. $\mathtt{c} : \mathtt{s}/\mathtt{s}$).*

For every default theory (D, W) that is singular, it straightforward to identify a subset $D' \subseteq D$ such that D' is a minimal set of defaults for each that extension of $D, W)$ is the same as the extension of (D', W). In this case, every default in $\alpha : \beta/\gamma \in D$ contributes to the extension (i.e. $\gamma \in E$), as illustrated in the following example.

Example 10. *Continuing Example 9, let $D' = \{\mathtt{a} : \mathtt{p}/\mathtt{q}\}$. So $D' \subseteq D$ and every default in D' contributes to the extension (i.e. $\mathtt{p}$ is in the extension of (D', W)). The extension of (D', W) is $\mathsf{Cn}(\{\mathtt{a}, \mathtt{b}, \mathtt{q}\})$, and there is no subset D'' of D such that the extension of (D'', W) is $\mathsf{Cn}(\{\mathtt{a}, \mathtt{b}, \mathtt{q}\})$.*

Justifications from different default rules do not need to be consistent together for an extension to exist as illustrated by the following example. Note, in the situations where this kind of reasoning arises, we can consider counterarguments that can attack the reasoning (as we shall investigate later in the paper).

Example 11. *Let* $D = \{\mathtt{a} : \mathtt{p}/\mathtt{q}, \mathtt{b} : \neg\mathtt{p}/\mathtt{r}\}$ *and* $W = \{\mathtt{a}, \mathtt{b}\}$. *Therefore,* (D, W) *is singular with the extension being* $\mathsf{Cn}(\{\mathtt{a}, \mathtt{b}, \mathtt{q}, \mathtt{r}\})$.

Some default theories have no extension as illustrated by the following example, and as such are not singular.

Example 12. *(Example 2 in [FM94]) Let* $D = \{\top : \mathtt{a}/\neg\mathtt{a}\}$ *and* $W = \emptyset$. *There is no extension containing* $\neg\mathtt{a}$ *that is consistent with* $\mathtt{a}$.

We will use the definition of singular default theories in the definition of default argument that we introduce in the next section.

4 Default arguments

We use the singular criteria (Definition 4) in the following definition of a default argument to ensure that the implicit premises (respectively implicit claim) give a single perspective on the explicit premises (respectively explicit claim). The definition is very general, and we will consider constraints on this definition in order to give us appropriate notions of logical argument.

Definition 5. *A* **default argument** *is a tuple* $\langle W^{\mathsf{p}}, D^{\mathsf{p}}, W^{\mathsf{c}}, D^{\mathsf{c}}\rangle$ *where* $W^{\mathsf{p}}, W^{\mathsf{c}} \subseteq \mathcal{L}$, *and* $D^{\mathsf{p}}, D^{\mathsf{c}} \subseteq \mathcal{D}$ *s.t.* $(D^{\mathsf{p}}, W^{\mathsf{p}})$ *is singular and* $(D^{\mathsf{c}}, W^{\mathsf{c}})$ *is singular.*

For a default argument $A = \langle W^{\mathsf{p}}, D^{\mathsf{p}}, W^{\mathsf{c}}, D^{\mathsf{c}}\rangle$, we refer to W^{p} as the **explicit premises**, D^{p} as the **implicit premises**, W^{c} as the **explicit claims**, and D^{c} as the **implicit claims**. To extract these components of a default argument A, we use the following functions: $\mathsf{Ep}(A) = W^{\mathsf{p}}$; $\mathsf{Ip}(A) = D^{\mathsf{p}}$; $\mathsf{Ec}(A) = W^{\mathsf{c}}$; and $\mathsf{Ic}(A) = D^{\mathsf{c}}$.

Example 13. *The following are examples of default arguments:*

$$\mathtt{A} = \langle\{\mathtt{a} \vee \mathtt{b}\}, \{(\mathtt{a} \vee \mathtt{b} \vee \mathtt{c} : \mathtt{d}/\mathtt{d})\}, \{\mathtt{d}\}, \{(\mathtt{d} : \neg\mathtt{e}/\neg\mathtt{e})\}\rangle.$$
$$\mathtt{B} = \langle\emptyset, \emptyset, \{\mathtt{b} \vee \neg\mathtt{b}\}, \emptyset\rangle.$$
$$\mathtt{C} = \langle\emptyset, \{(: \mathtt{e}/\mathtt{e})\}, \emptyset, \{(\mathtt{e} : \mathtt{f}/\mathtt{f}), (\mathtt{f} : \mathtt{g} \wedge \mathtt{h}/\mathtt{h})\}\rangle.$$

As we will explore in the rest of this paper, a default argument provides a richer representation of an argument than available with other approaches to structured argumentation. This includes the following features which go beyond other formalisms for logic-based argumentation: **Delineatation of implicit information connecting premises and claims:** The set D^{p} is a set of defaults that represents the implicit information in the premises; **Logical mechanism for disabling connection between premises and claims:** The justification of each default rule can be negated by the claim of another argument, thereby attacking the connection between the premise and claim; **Delineatation of implicit information connecting one argument with another:** The set D^{c} is a set of defaults that represents the implicit information in the claim; **Logical mechanism for disabling connection between one argument and another:** The justification of each default rule can be negated by the claim of another argument, thereby attacking the connection between that argument and other arguments.

We give further examples of default arguments in each node in the instantiated argument maps in Figure 6, Figure 7, and Figure 8: Figure 6 provides a straightforward and common kind of example to illustrate how it can be handled in our approach; and Figure 7 captures the argument map from Figure 2, and it shows the value of implicit claims — in particular, it shows how implict claims capture how one enthymeme attacks or supports another enthymeme. We will explain the instantiation process used in these examples in more detail in later sections including the nature of attacks and support for default arguments.

Let $\mathcal{R}$ be the set of all possible default rules. The **set of default arguments** from a set of default rules $\Pi \subseteq \mathcal{R}$ is $\mathsf{Arguments}(\Pi) = \{\langle W^{\mathsf{p}}, D^{\mathsf{p}}, W^{\mathsf{c}}, D^{\mathsf{c}}\rangle \mid W^{\mathsf{p}}, W^{\mathsf{c}} \subseteq \mathcal{L}$ and $D^{\mathsf{p}}, D^{\mathsf{c}} \subseteq \Pi\}$. Using default logic

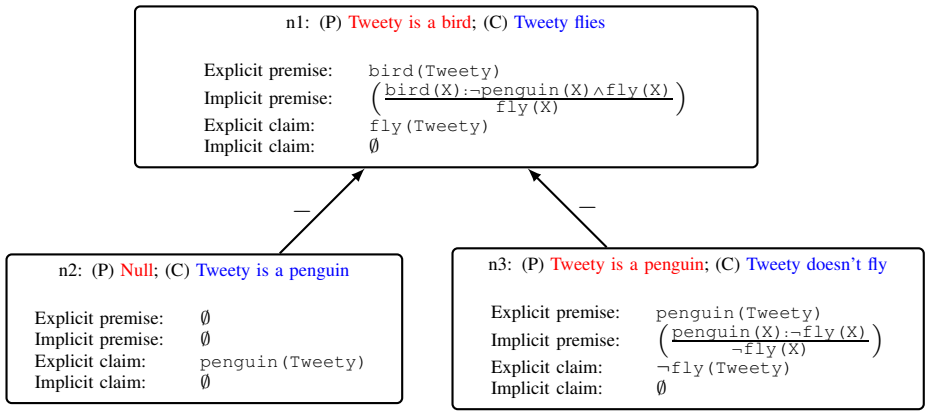


Figure 6: An example of an instantiated argument map concerning birds flying and penguins not flying. Here, there is no implicit claim in any of these arguments. We explain these default arguments as follows: Argument at n1 has an explicit premise of $\texttt{bird(Tweety)}$ which satisfies the pre-condition for the default rule in the implicit premise, and this gives the explicit claim $\texttt{fly(Tweety)}$; Argument at n2 has the explicit claim of $\texttt{penguin(Tweety)}$; and Argument at n3 has an explicit premise of $\texttt{penguin(Tweety)}$ which satisfies the pre-condition for the default rule in the implicit premise, and this gives the explicit claim $\neg\texttt{fly(Tweety)}$. The explicit claim of arguments at n2 and n3 negate the justification condition for the default rule in the argument in node n1. Also, the claim of the argument at n3 negates the claim of the argument at n2.

as a base logic in this way does not affect the argument construction being monotonic; Adding a formula to the knowledge-base would not cause a default argument to be withdrawn. Rather it may allow further default arguments to be constructed. So if $\Pi \subseteq \Pi'$, then $\mathsf{Arguments}(\Pi) \subseteq \mathsf{Arguments}(\Pi')$.

Given a default argument A, the support of the argument is the default extension obtained from the implicit and explicit premises, and the consequence of the argument is the default extension obtained from the implicit and explicit claim:

- The **support** of A is $\mathsf{S}(A) = \mathsf{Ex}(\mathsf{Ip}(A), \mathsf{Ep}(A))$.
- The **consequence** of A is $\mathsf{C}(A) = \mathsf{Ex}(\mathsf{Ic}(A), \mathsf{Ec}(A))$.

So the support of a default argument captures the consequences of the implicit and explicit premises and the claim of a default argument captures the consequences of the implicit and explicit claims.

Example 14. *Continuing Example 13, for the default argument* $\mathtt{A}$ *to* $\mathtt{C}$*, we have the following support and consequence.*

$$\begin{array}{lll}
\mathtt{A} = \langle\{\mathtt{a} \vee \mathtt{b}\}, \{(\mathtt{a} \vee \mathtt{b} \vee \mathtt{c} : \mathtt{d}/\mathtt{d})\}, \{\mathtt{d}\}, \{(\mathtt{d} : \neg\mathtt{e}/\neg\mathtt{e})\}\rangle & \mathsf{S}(\mathtt{A}) = \mathsf{Cn}(\{\mathtt{a} \vee \mathtt{b}, \mathtt{d}\}) & \mathsf{C}(\mathtt{A}) = \mathsf{Cn}(\{\mathtt{d}, \neg\mathtt{e}\}) \\
\mathtt{B} = \langle\emptyset, \emptyset, \{\mathtt{b} \vee \neg\mathtt{b}\}, \emptyset\rangle & \mathsf{S}(\mathtt{B}) = \mathsf{Cn}(\emptyset) & \mathsf{C}(\mathtt{B}) = \mathsf{Cn}(\emptyset) \\
\mathtt{C} = \langle\emptyset, \{(\top : \mathtt{e}/\mathtt{e}), (\mathtt{f} : \mathtt{g} \wedge \mathtt{h}/\mathtt{h})\}, \{\mathtt{h}\}, \{(\mathtt{h} : \mathtt{f}/\mathtt{f})\}\rangle & \mathsf{S}(\mathtt{C}) = \mathsf{Cn}(\{\mathtt{e}\}) & \mathsf{C}(\mathtt{C}) = \mathsf{Cn}(\{\mathtt{f}, \mathtt{h}\})
\end{array}$$

The definition for a default argument is very general, and so we consider some types of argument including the following.

Definition 6. *For a default argument* A*,*

- A *is* **valid** *iff* $\mathsf{Ec}(A) \subseteq \mathsf{S}(A)$.
- A *is* **implicitly minimal** *iff* A *is valid and there is no* $\Phi \subset \mathsf{Ip}(A)$ *s.t.* $\mathsf{Ec}(A) \subseteq \mathsf{Ex}(\Phi, \mathsf{Ep}(A))$.
- A *is* **explicitly minimal** *iff* A *is valid and there is no* $\Psi \subset \mathsf{Ep}(A)$ *s.t.* $\mathsf{Ec}(A) \subseteq \mathsf{Ex}(\mathsf{Ip}(A), \Psi)$.
- A *is* **support consistent** *iff* $\bot \notin \mathsf{S}(A)$.
- A *is* **consequence consistent** *iff* $\bot \notin \mathsf{C}(A)$.
- A *is* **fully consistent** *iff* $\mathsf{S}(A) \cup \mathsf{C}(A) \nvdash \bot$.

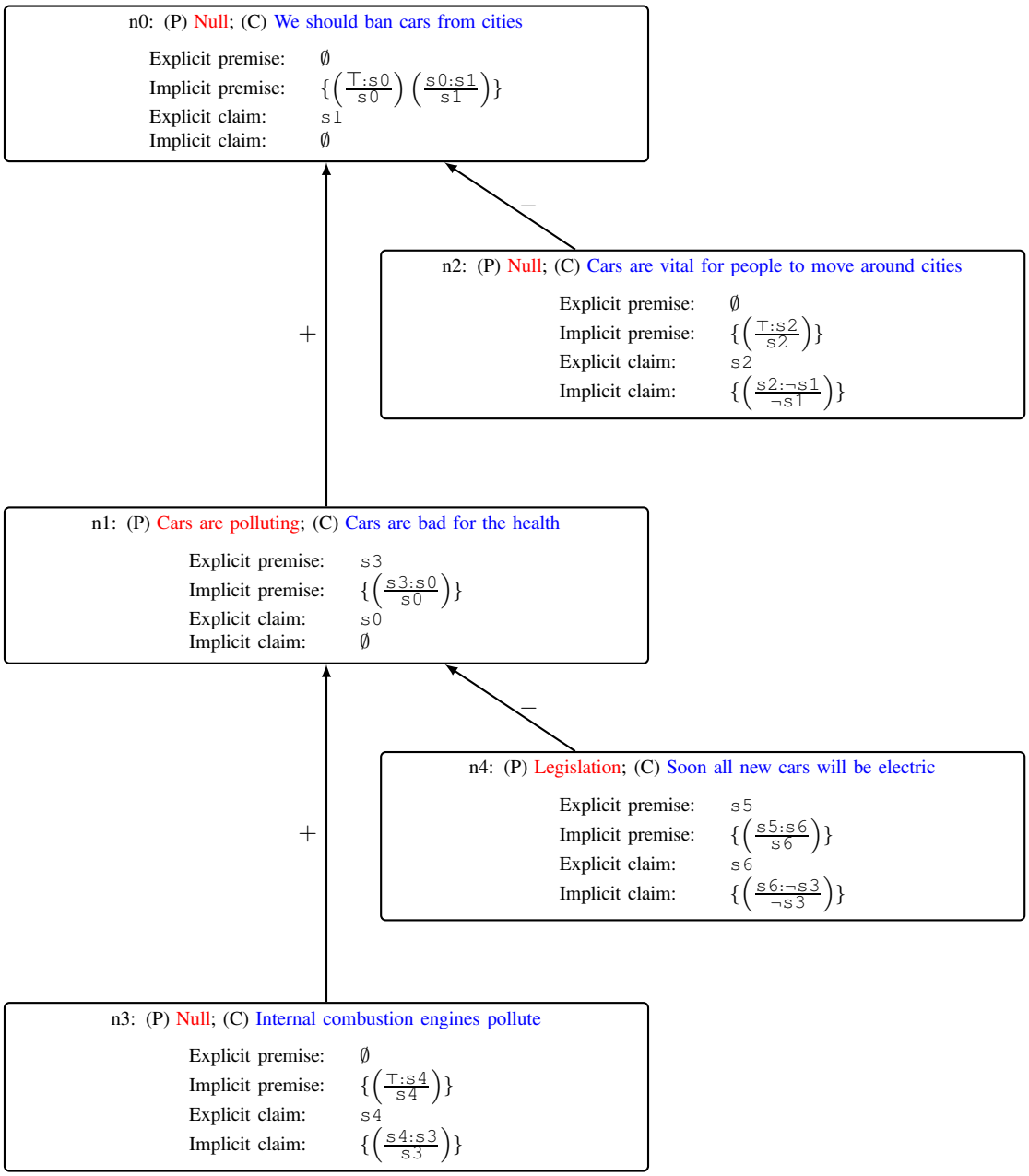
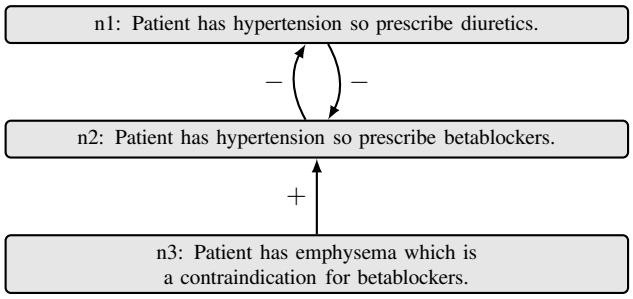
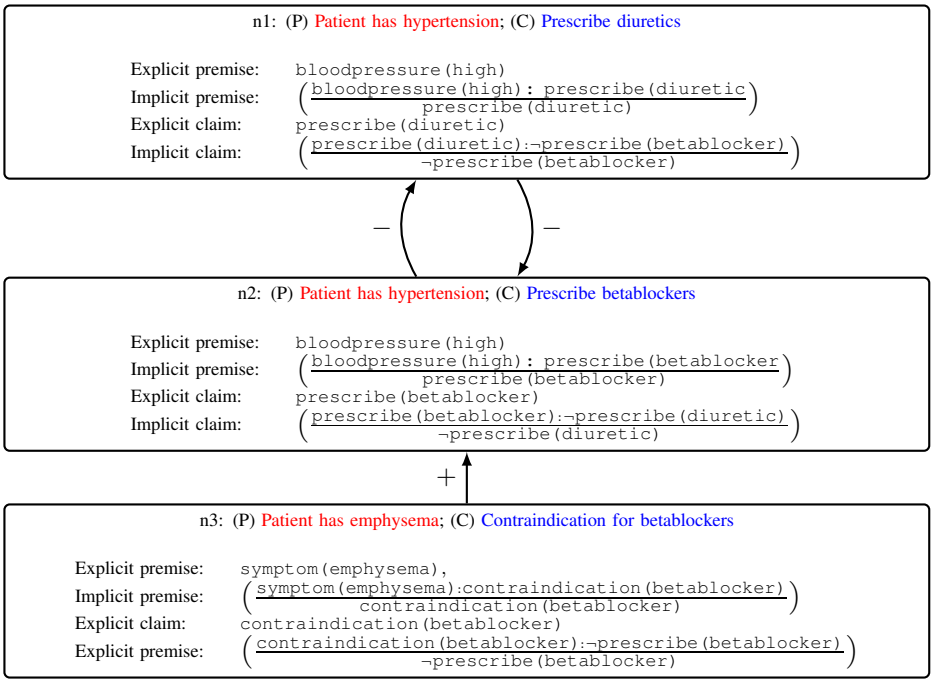


Figure 7: An instantiated argument map for the argument map in Figure 2. The atoms are s0 = *cars are bad for health*, s1 = *Cars should be banned from cities*, s2 = *Cars are vital for people to move around cities*, s3 = *Cars are polluting*, s4 = *Internal combustion engines pollute*, s5 = *Legislation*, and s6 = *Soon all new cars will be electric*. We explain the structuring of the arguments at each node as follows: (n0) argument has implicit premises that imply the explicit claim s1; (n1) argument has the explicit premise s3, which is used to derive the explicit claim s0, and this is a support for argument at n0; (n2) argument has the implicit premise that is used to derive the explicit claim s2, and this is then used with the implicit claim to give ¬s1, and this constitutes an attack on argument at n0; (n3) argument has implicit premises that is used to derive s3, and this is then used as a support for the explicit premises of the argument at n1; and (n4) argument has the explicit claim ¬s3, and this is then used as an attack on the explicit premise of the argument at n1.

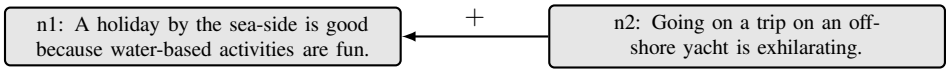


(a) An argument graph that captures the arguments and counterarguments in a decision making scenario.

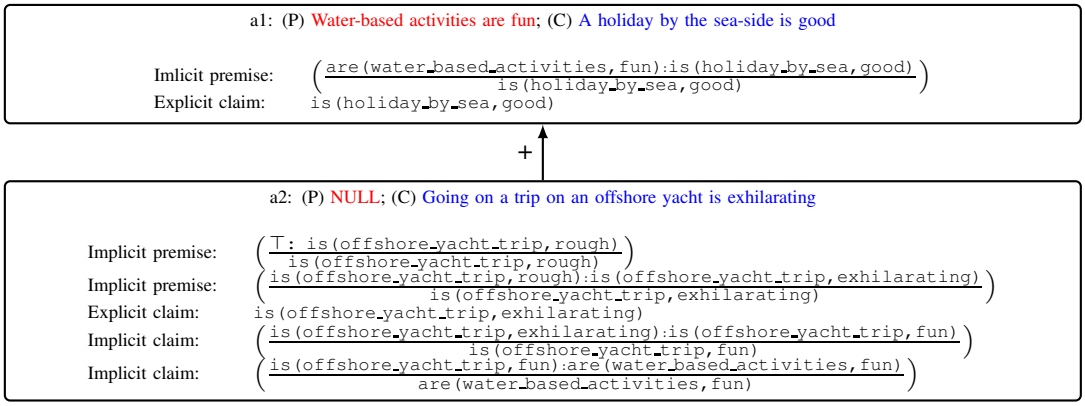


(b) An instantiated argument map generated from the argument graph in Figure 8a.

Figure 8: A decision making scenario where there are two alternatives for treating a patient, diuretics or betablockers. Since only one treatment should be given for the disorder, each argument attacks the other. There is also a reason to not give betablockers, as the patient has emphysema which is a contraindication for this treatment.



(a) An argument graph where the argument at n2 is a supporting argument for the argument at n1.



(b) An instantiated argument map generated from the argument map in Figure 9a.

Figure 9: An example of an instantiated argument map in Figure 9b generated from the argument map, concerning a type of holiday, in Figure 9a. In this example, we see the use of the implicit claim to generate an inference in the consequence of the supporting argument that is a fact in the premises of the supported argument.

We explain the above definitions as follows: A default argument is valid iff the explicit claim is in the support of the argument (i.e. the extension of the premises); A default argument is implicitly minimal iff there is no subset of the implicit premises that is valid; A default argument is explicitly minimal iff there is no subset of the explicit premises that is valid; A default argument is support consistent iff the support is consistent; A default argument is claim consistent iff the claim is consistent; And a default argument is fully consistent iff the support and claim are consistent together. Clearly, if an argument is fully coherent, then it is support consistent and claim consistent.

Example 15. *Continuing Example 13, default arguments* $\mathtt{A}$*.* $\mathtt{B}$*, and* $\mathtt{C}$*, are valid, fully consistent, and minimal, whereas* $\mathtt{D}$ *is valid and fully consistent but not minimal.*

$$\mathtt{A} = \langle\{\mathtt{a} \vee \mathtt{b}\}, \{(\mathtt{a} \vee \mathtt{b} \vee \mathtt{c} : \mathtt{d}/\mathtt{d})\}, \{\mathtt{d}\}, \{(\mathtt{d} : \neg\mathtt{e}/\neg\mathtt{e})\}\rangle.$$
$$\mathtt{B} = \langle\emptyset, \emptyset, \{\mathtt{b} \vee \neg\mathtt{b}\}, \emptyset\rangle.$$
$$\mathtt{C} = \langle\emptyset, \{(: \mathtt{e}/\mathtt{e})\}, \{\mathtt{e}\}, \{(\mathtt{e} : \mathtt{f}/\mathtt{f}), (\mathtt{f} : \mathtt{g} \wedge \mathtt{h}/\mathtt{h})\}\rangle.$$
$$\mathtt{D} = \langle\emptyset, \{(: \mathtt{e}/\mathtt{e})\}, \emptyset, \{(\mathtt{e} : \mathtt{f}/\mathtt{f}), (\mathtt{f} : \mathtt{g} \wedge \mathtt{h}/\mathtt{h})\}\rangle.$$

Example 16. *The default arguments* $\mathtt{A} = \langle\{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{b})\}, \{\mathtt{b}\}, \{(\mathtt{b} : \neg\mathtt{a}/\neg\mathtt{a})\}\rangle$ *and* $\mathtt{B} = \langle\{\mathtt{a}\}, \emptyset, \{\neg\mathtt{a}\}, \emptyset\rangle$ *are support and consequence consistent (as shown below) but they are not fully consistent since* $\mathsf{S}(A) \cup \mathsf{C}(A) \vdash \bot$ *and* $\mathsf{S}(B) \cup \mathsf{C}(B) \vdash \bot$.

$$\mathsf{S}(A) = \mathsf{Cn}(\{\mathtt{a}, \mathtt{b}\}) \quad \mathsf{C}(A) = \mathsf{Cn}(\{\neg\mathtt{a}, \mathtt{b}\})$$
$$\mathsf{S}(B) = \mathsf{Cn}(\{\mathtt{a}\}) \quad \mathsf{C}(B) = \mathsf{Cn}(\{\neg\mathtt{a}\})$$

A default argument A is **explicit** iff $\mathsf{Ip}(A) = \mathsf{Ic}(A) = \emptyset$. If A is explicit, valid, and fully consistent, then for each $\beta \in \mathsf{Ec}(A)$, $\mathsf{Ep}(A) \vdash \beta$. A default argument A is a **classical argument** iff A is explicit, valid, explicitly minimal, and support consistent. So a classical argument is a minimal argument, and a classical argument is fully consistent. A default argument A is a **non-classical argument** iff A is not a classical argument.

Example 17. *The default argument* $\langle\{\mathtt{a} \vee \mathtt{b}\}, \{(\mathtt{a} \vee \mathtt{b} \vee \mathtt{c} : \mathtt{d}/\mathtt{d})\}, \{\mathtt{a} \vee \mathtt{b} \vee \mathtt{e}\}, \emptyset\rangle$ *is valid, explicitly minimal, and fully consistent, but it is not explicit nor implicitly minimal, and therefore it is not classical, whereas the default argument* $\langle\{\mathtt{a} \vee \mathtt{b}\}, \emptyset, \{\mathtt{a} \vee \mathtt{b} \vee \mathtt{e}\}, \emptyset\rangle$ *is explicit, explicitly minimal, valid, and fully consistent, and therefore classical. The following default arguments are non-classical:* $\langle\{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle$*,* $\langle\{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{c})\}, \{\mathtt{d}\}, \emptyset\rangle$*,* $\langle\{\mathtt{a}\}, \emptyset, \{\mathtt{d}\}, \emptyset\rangle$*, and* $\langle\{\mathtt{a} \wedge \neg\mathtt{a}\}, \emptyset, \{\mathtt{d}\}, \emptyset\rangle$

We refer to argument $\langle\emptyset, \emptyset, \emptyset, \emptyset\rangle$ as a **vacuous argument**. It is valid, implicitly minimal, explicitly minimal, fully consistent, and classical.

Example 18. *Continuing Example 13, the default argument* $\mathtt{B} = \langle\emptyset, \emptyset, \{\mathtt{b} \vee \neg\mathtt{b}\}, \emptyset\rangle$ *is vacuous.*

A default argument A has **completely implicit premises** iff $\mathsf{Ep}(A) = \emptyset$ and $\mathsf{Ip}(A) \neq \emptyset$; and A has a **completely implicit claims** iff $\mathsf{Ec}(A) = \emptyset$ and $\mathsf{Ic}(A) \neq \emptyset$.

Example 19. *Continuing Example 13, the argument* $\mathtt{C} = \langle\emptyset, \{(: \mathtt{e}/\mathtt{e})\}, \emptyset, \{(\mathtt{e} : \mathtt{f}/\mathtt{f}), (\mathtt{f} : \mathtt{g} \wedge \mathtt{h}/\mathtt{h})\}\rangle$ *has completely implicit premises and claims, whereas* $\mathtt{D} = \langle\emptyset, \{(: \mathtt{e}/\mathtt{e})\}, \{\mathtt{e}\}, \emptyset\rangle$ *has completely implicit premises and explicit claims.*

Proposition 3. *For all* $\Pi \subseteq \mathcal{R}$*, where* $\mathcal{R}$ *is the set of default rules, if* A *is a classical argument, then* $A \in \mathsf{Arguments}(\Pi)$.

Proof. If A is a classical argument, then it is of the form $\langle W^{\mathsf{p}}, D^{\mathsf{p}}, W^{\mathsf{c}}, D^{\mathsf{c}}\rangle$. Let $D^{\mathsf{p}} = D^{\mathsf{c}} = \emptyset$, and let W^{p} and W^{c} are such that for each $\beta \in W^{\mathsf{c}}$, $W^{\mathsf{p}} \vdash \beta$. So $A \in \mathsf{Arguments}(\Pi)$. □

Default logic does not allow an inconsistent extension except when the premise is inconsistent. So an inconsistent claim arises only if the explicit premise is inconsistent.

Proposition 4. *If default argument* A *is a valid explicit argument, and* $\mathsf{Ec}(A) \vdash \bot$*, then* $\bot \in \mathsf{Ep}(A)$.

Proof. Assume A is explicit. So $\mathsf{Ip}(A) = \emptyset$ and $\mathsf{Ic}(A) = \emptyset$. Also assume A is valid. So $\mathsf{Ec}(A) \subseteq \mathsf{Ex}(\mathsf{Ip}(A), \mathsf{Ep}(A))$. Recall that $\mathsf{Ec}(A) \vdash \bot$ is assumed. Now the only way for $\bot \in \mathsf{Ex}(\mathsf{Ip}(A), \mathsf{Ep}(A))$ to hold, is for $\bot \in \mathsf{Ep}(A)$ to hold. □

For a default argument A, if neither $\mathsf{Ep}(A)$ nor $\mathsf{Ec}(A)$ is contradictory, then A is consistent, as captured by the following result.

Proposition 5. *For default argument A, (1) if $\bot \in \mathsf{S}(A)$, then $\mathsf{Ep}(A) \equiv \bot$; (2) if $\bot \in \mathsf{C}(A)$, then $\mathsf{Ec}(A) \equiv \bot$.*

Proof. (1) Assume $\bot \in \mathsf{S}(A)$. So $\mathsf{Ex}(\mathsf{Ip}(A), \mathsf{Ep}(A))$ is inconsistent. But this is only possible if $\mathsf{Ep}(A)$ is inconsistent. So $\mathsf{Ep}(A) \equiv \bot$ holds. (2) Assume $\bot \in \mathsf{C}(A)$. So $\mathsf{Ex}(\mathsf{Ic}(A), \mathsf{Ec}(A))$ is inconsistent. But this is only possible if $\mathsf{Ec}(A)$ is inconsistent. So $\mathsf{Ec}(A) \equiv \bot$ holds. □

Example 20. *Consider the following arguments.*

- $A = \langle \{\mathtt{a} \wedge \neg\mathtt{a}\}, \emptyset, \{\mathtt{d}\}, \{\mathtt{d} : \mathtt{e}/\mathtt{e}\}\rangle$ *is such that* $\bot \in \mathsf{S}(A)$ *and* $\bot \notin \mathsf{C}(A)$.
- $B = \langle \{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{b}\}, \{\mathtt{d} \wedge \neg\mathtt{d}\}, \emptyset\rangle$ *is such that* $\bot \notin \mathsf{S}(B)$ *and* $\bot \in \mathsf{C}(B)$.

With the richer structure that comes with default arguments, there are various ways that two arguments can be formalized as equivalent.

Definition 7. *Types of equivalence between default arguments A and B include:*

- *A is* **explicitly equivalent** *to B iff* $\mathsf{Cn}(\mathsf{Ep}(A)) = \mathsf{Cn}(\mathsf{Ep}(B))$ *and* $\mathsf{Cn}(\mathsf{Ec}(A)) = \mathsf{Cn}(\mathsf{Ec}(B))$.
- *A is* **support equivalent** *to B iff* $\mathsf{S}(A) = \mathsf{S}(B)$.
- *A is* **consequence equivalent** *to B iff* $\mathsf{C}(A) = \mathsf{C}(B)$.
- *A is* **implicitly equivalent** *to B iff A is support equivalent and consequence equivalent to B.*
- *A is* **intrinsically equivalent** *to B iff A is support equivalent to B and* $\mathsf{Cn}(\mathsf{Ec}(A)) = \mathsf{Cn}(\mathsf{Ec}(B))$.

We explain the above definitions as follows: A is explicitly equivalent to B when the explicit premises are equivalent and the explicit claims are equivalent; A is support equivalent to B when the supports are the same; A is claim equivalent to B when the claims are the same; and A is implicitly equivalent to B when they are both premise and claim equivalent. Since the implicitness of an argument's claim depends on its relations to another argument, A is intrinsically equivalent to B when they are both equivalent regardless of their relations.

Example 21. *Default arguments* `A`*,* `B` *and* `C` *are implicitly equivalent.*

$$\mathtt{A} = \langle \{\mathtt{e}\}, \{(\mathtt{e} : \mathtt{f}/\mathtt{f})\}, \{\mathtt{f}\}, \{(\mathtt{f} : \neg\mathtt{b}/\neg\mathtt{b})\}\rangle.$$
$$\mathtt{B} = \langle \{\mathtt{e}\}, \{(\mathtt{e} : \mathtt{f}/\mathtt{f})\}, \{\mathtt{f} \wedge \neg\mathtt{b}\}, \emptyset\rangle.$$
$$\mathtt{C} = \langle \emptyset, \{(: \mathtt{e}/\mathtt{e}), (\mathtt{e} : \mathtt{f}/\mathtt{f})\}, \{\neg(\neg\mathtt{f} \vee \mathtt{b})\}, \emptyset\rangle.$$

Proposition 6. *For every classical (respectively valid, implicitly minimal, and explicitly minimal, non-classical) argument A, there is a valid, implicitly minimal, and explicitly minimal, non-classical (respectively classical) argument B, s.t. A is implicitly equivalent to B.*

Proof. Let A be a classical argument. So $\mathsf{Ip}(A) = \emptyset$ and $\mathsf{Ic}(A) = \emptyset$. Let B be a valid, implicitly minimal, and explicitly minimal, non-classical argument. So $\mathsf{Ip}(B) \neq \emptyset$ or $\mathsf{Ic}(B) \neq \emptyset$. Therefore, for each A, a B can be chosen so that $\mathsf{S}(A) \equiv \mathsf{S}(B)$ and $\mathsf{C}(A) \equiv \mathsf{C}(B)$, and hence so that A is implicitly equivalent to B. In the same way, for each B, an A can be chosen so that they are implicitly equivalent. □

We can also rank arguments by their implicitness: A is more implicit than B when they are implicitly equivalent but the explicit premises and claim of A are weaker than those of B.

- A is **more implicit** than B iff A is implicitly equivalent to B and $\mathsf{Cn}(\mathsf{Ep}(A)) \subseteq \mathsf{Cn}(\mathsf{Ep}(B))$ and $\mathsf{Cn}(\mathsf{Ec}(A)) \subseteq \mathsf{Cn}(\mathsf{Ec}(B))$.

To conclude this section, default arguments are a general kind of argument, with some important special cases, that meet our needs for formalizing textual arguments as arising in argument maps. The advantage of using default logic in arguments is that it allows default inferences to be drawn. This means that we use a well-developed and well-understand formalism for representing and reasoning with the complexities of default knowledge. Hence, we can have a richer and more natural representation of defaults. It also means that inferences can be drawn in the absence of reasons to not draw them. For instance, we can conclude the something flies from knowing that is a bird in the absence of knowing whether it is a penguin. In other words, we just need to know that it is consistent to believe that it is not a penguin and that it is consistent to believe that it can fly. Furthermore, we just need to do this consistency check within the premises of the argument. Note, this consistency check is different to the consistency check used for default negation in DeLP which involves checking consistency with all the strict knowledge (i.e. the subset of knowledge that is assumed to be correct) [GS04].

5 Relationships between arguments

We now consider how one default argument supports or attacks another default argument. For this, we will require some subsidiary definitions. The first is to identify the justifications that arise in the default rules in the premises of an argument using the following definition for the **premise justification function**, where A is an argument.

$$\mathsf{Jp}(A) = \{\beta \mid \alpha : \beta/\gamma \in \mathsf{Ip}(A)\}$$

Example 22. *For* $\mathtt{A} = \langle\{\mathtt{e}\}, \{(\mathtt{e} : \mathtt{f} \wedge \mathtt{a}/\mathtt{f})\}, \{\mathtt{f}\}, \{(\mathtt{f} : \neg\mathtt{b} \wedge (\mathtt{a} \vee \mathtt{c})/\neg\mathtt{b})\}\rangle$, $\mathsf{Jp}(\mathtt{A}) = \{\mathtt{f} \wedge \mathtt{a}\}$.

We also require the following variants of the S and C functions. For any default argument A, $\mathsf{C}^*(A) = \mathsf{C}(A) \setminus \mathsf{Cn}(\{\top\})$ and $\mathsf{S}^*(A) = \mathsf{S}(A) \setminus \mathsf{Cn}(\{\top\})$. We use $\mathsf{C}^*(A)$ and $\mathsf{S}^*(A)$ instead of $\mathsf{C}(A)$ and $\mathsf{S}(A)$ as we want to avoid trivial support involving tautologies.

We define the following notions of support relation that hold between a pair of default arguments. Whilst there are further kinds of support that we could define, we start with these intuitive options. Essentially, we restrict the definitions to comparing the explicit claim or consequence (i.e. extension of the implicit and explicit claims) of a supporting argument A (i.e. $\mathsf{Ec}(A)$ or $\mathsf{C}(A)$), and the explicit premises, or support (i.e. extension of the implicit and explicit premises), or justification, of the supported argument B (i.e. $\mathsf{Ep}(B)$, or $\mathsf{S}(B)$, or $\mathsf{Jp}(B)$), and we restrict the comparison between the supporting and supported arguments to an intersection between the respective sets of formulae.

Definition 8. *For default arguments A and B, the following are types of support.*

- A **inferentially supports** B *iff* $\mathsf{S}^*(B) \cap \mathsf{C}^*(A) \neq \emptyset$.
- A **directly supports** B *iff* $\mathsf{Ep}(B) \cap \mathsf{C}^*(A) \neq \emptyset$.
- A **explicitly supports** B *iff* $\mathsf{Ep}(B) \cap \mathsf{Cn}(\mathsf{Ec}(A)) \neq \emptyset$.
- A **justification supports** B *iff* $\mathsf{Jp}(B) \cap \mathsf{C}^*(A) \neq \emptyset$.

We explain the above definitions as follows and provide an example below: Inferential support ensures that there is a consequence of A that is a support in B; Direct support ensures that there is a consequence of A that is an explicit premise in B; Explicit support ensures that the explicit claim in A implies an explicit premise in B; And justification support ensures that there is a consequence of A that is a justification in the implicit premises in B.

Example 23. *Consider the following supporting default arguments (left) and supported default arguments (right):* A1 *inferentially, directly, but not explicitly, supports* B1. A2 *inferentially, directly, and explicitly, supports* B2. A3 *justification, but not inferentially, supports* B3. A4 *justification, inferentially, directly, and explicitly, supports* B4.

$$\begin{array}{ll} \mathtt{A1} = \langle\emptyset, \emptyset, \{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{b})\}\rangle & \mathtt{B1} = \langle\{\mathtt{b}\}, \{(\mathtt{b} : \mathtt{c}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle \\ \mathtt{A2} = \langle\emptyset, \emptyset, \{\mathtt{a}\}, \emptyset\rangle & \mathtt{B2} = \langle\{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle \\ \mathtt{A3} = \langle\emptyset, \emptyset, \{\mathtt{b}\}, \emptyset\rangle & \mathtt{B3} = \langle\{\mathtt{a}\}, \{(\mathtt{b} : \mathtt{c}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle \\ \mathtt{A4} = \langle\emptyset, \emptyset, \{\mathtt{a} \wedge \mathtt{b}\}, \emptyset\rangle & \mathtt{B4} = \langle\{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle \end{array}$$

For each form of support above, we provide a more restricted form of the support. Essentially, above the relationship between the formulae considered in the supported argument and supporting argument is intersection, whereas below we replace intersection with the subset or equal relation. In each definition, we recall the more general definition to ensure that there is a non-empty set of formulae considered for the supported and supporting arguments, otherwise we may have trivial support. For example, for inferentially supports, if $\mathsf{S}(B)$ is the emptyset, and we did not specify that A inferentially supports B holds, then we would have that A strongly inferentially supports B, and so we would have trivial form of support. Ensuring that A inferentially supports B holds obviates this possibility.

Definition 9. *For default arguments A and B, the following are types of support.*

- A **strongly inferentially supports** B *iff* A *inferentially supports* B *and* $\mathsf{S}(B) \subseteq \mathsf{C}(A)$.
- A **strongly directly supports** B *iff* A *directly supports* B *and* $\mathsf{Ep}(B) \subseteq \mathsf{C}(A)$.
- A **strongly explicitly supports** B *iff* A *explicitly supports* B *and* $\mathsf{Ep}(B) \subseteq \mathsf{Cn}(\mathsf{Ec}(A))$.
- A **strongly justification supports** B *iff* A *justification supports* B *and* $\mathsf{Jp}(B) \subseteq \mathsf{C}(A)$.

Example 24. *Consider the following supporting default arguments (left) and supported default arguments (right):* A1 *strongly inferentially, but not strongly directly, nor strongly explicitly, supports* B1. A2 *strongly directly, but not strongly explicitly, nor strongly inferentially, supports* B2. A3 *strongly directly, and strongly explicitly, but not strongly inferentially, supports* B3. A4 *strongly inferentially, strongly directly, and strongly explicitly, supports* B4. *and* A5 *strongly justification, but not strongly inferentially, nor strongly directly, nor strongly explicitly, supports* B4.

$$\begin{array}{ll}
\mathtt{A1} = \langle \emptyset, \emptyset, \{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{b})\}\rangle & \mathtt{B1} = \langle \emptyset, \{(\top : \mathtt{b}/\mathtt{b})\}, \{\mathtt{b}\}, \emptyset\rangle \\
\mathtt{A2} = \langle \emptyset, \emptyset, \{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{b})\}\rangle & \mathtt{B2} = \langle \{\mathtt{b}\}, \{(\mathtt{b} : \mathtt{c}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle \\
\mathtt{A3} = \langle \emptyset, \emptyset, \{\mathtt{a}\}, \emptyset\rangle & \mathtt{B3} = \langle \{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{c})\}, \{\mathtt{c}\}, \emptyset\rangle \\
\mathtt{A4} = \langle \{\mathtt{d}\}, \{\mathtt{d} : \mathtt{a}/\mathtt{a}\}, \{\mathtt{a}\}, \emptyset\rangle & \mathtt{B4} = \langle \{\mathtt{a}\}, \emptyset, \{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{c}\}\rangle \\
\mathtt{A5} = \langle \{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{c}\}, \{\mathtt{c}\}, \emptyset\rangle & \mathtt{B5} = \langle \{\mathtt{d}\}, \{\mathtt{d} : \mathtt{c}/\mathtt{e}\}, \{\mathtt{e}\}, \emptyset\rangle
\end{array}$$

An inconsistent default argument, is inferentially supported by, and inferentially supports, any default argument.

Example 25. *Consider* $\mathtt{A} = \langle \{\bot\}, \emptyset, \{\bot\}, \emptyset\rangle$, *and* $\mathtt{B} = \langle \{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{b}/\mathtt{b})\}, \{\mathtt{b}\}, \{(\mathtt{b} : \mathtt{c}/\mathtt{c})\}\rangle$. *So* A *inferentially supports, directly supports, explicitly supports, and justification supports,* B, *and* B *inferentially supports, but not directly suppports, nor explicitly supports, nor justification supports* A.

Proposition 7. *If default argument A directly supports, or explicitly supports, default argument B, then A inferentially supports B.*

Proof. (1) Assume A directly supports B. So $\mathsf{Ep}(B) \in \mathsf{C}(A)$. Since $\mathsf{Ep}(B) \in \mathsf{S}(B)$, $\mathsf{C}(A) \cap \mathsf{S}(B) \neq \emptyset$. (2) Assume A explicitly supports B. So $\mathsf{Cn}(\mathsf{Ec}(A)) \cap \mathsf{S}(B) \neq \emptyset$. So $\mathsf{C}(A) \cap \mathsf{S}(B) \neq \emptyset$. □

Now we turn to notions of attack from one argument on another. We use the following definition of attacks for a default argument A on default argument B which essentially specifies attack as being an inconsistency between the claim of the attacker and either the support or claim of the attackee. Another kind of attack involves inconsistency between the support of an argument and the justification of the the other argument. In this attacker, the attacker negates the justification of the attackee, and thereby presents a reason to block the use of the default rule.

Definition 10. *For default arguments A and B, the following are types of attack.*

- A **support attacks** B *iff* $\mathsf{C}(A) \cup \mathsf{S}(B) \vdash \bot$.
- A **consequence attacks** B *iff* $\mathsf{C}(A) \cup \mathsf{C}(B) \vdash \bot$.
- A **justification attacks** B *iff* $\mathsf{C}(A) \cup \mathsf{Jp}(B)) \vdash \bot$.

Example 26. *For the arguments below, we have the following relationships:* A *support attacks* B, A *consequence attacks* B, C *support attacks* D, C *does not claim attack* D, E *does not support attack* F, E *consequence*

Aspect of A that is attacked		Attack relation on A	Type of attack on A
Explicit premise	$\mathsf{Ep}(A)$	Undermine	Support attack
Default premise	$\mathsf{Ip}(A)$	Undercut	Justification attack
Explicit claim	$\mathsf{Ec}(A)$	Rebut	Consequence attack
Default claim	$\mathsf{Ic}(A)$	Overcut	Justification attack

Table 1: Types of focused attack on a default argument A according the aspect of A that is attacked. For instance, undermine is an attack on the explicit premise.

attacks $\mathtt{F}$*,* $\mathtt{G}$ *justification attacks* $\mathtt{H}$*, and* $\mathtt{I}$ *justification attacks* $\mathtt{J}$*.*

$$\mathtt{A} = \langle\{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{b}\}, \{\mathtt{b}\}, \{\mathtt{b} : \neg\mathtt{e}/\neg\mathtt{e}\}\rangle \qquad \mathtt{B} = \langle\{\mathtt{d}\}, \{\mathtt{d} : \mathtt{c} \wedge \mathtt{e}/\mathtt{e}\}, \{\mathtt{e}\}, \emptyset\rangle$$
$$\mathtt{C} = \langle\{\mathtt{a}\}, \emptyset, \{\mathtt{a}\}, \emptyset\rangle \qquad \mathtt{D} = \langle\emptyset, \{\top : \neg\mathtt{a}/\neg\mathtt{a}, \neg\mathtt{a} : \mathtt{b}/\mathtt{b}\}, \{\mathtt{b}\}, \emptyset\rangle$$
$$\mathtt{E} = \langle\{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b} \wedge \mathtt{c}/\mathtt{c}\}, \{\mathtt{c}\}, \emptyset\rangle \qquad \mathtt{F} = \langle\{\mathtt{e}\}, \{\mathtt{e} : \mathtt{f}/\mathtt{f}\}, \{\mathtt{f}\}, \{\mathtt{f} : \neg\mathtt{c}/\neg\mathtt{c}\}\rangle$$
$$\mathtt{G} = \langle\{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{b}\}, \{\mathtt{b}\}, \{\mathtt{b} : \neg\mathtt{c}/\neg\mathtt{c}\}\rangle \qquad \mathtt{H} = \langle\{\mathtt{d}\}, \{\mathtt{d} : \mathtt{c} \wedge \mathtt{e}/\mathtt{e}\}, \{\mathtt{e}\}, \emptyset\rangle$$
$$\mathtt{I} = \langle\{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{b}\}, \{\mathtt{b}\}, \emptyset\rangle \qquad \mathtt{J} = \langle\{\mathtt{d}\}, \{\mathtt{d} : \neg\mathtt{b} \wedge \mathtt{e}/\mathtt{e}\}, \{\mathtt{e}\}, \emptyset\rangle$$

So a justification attack by an argument negates a justification of default rule used in the attacked argument. Note, there is no link between support attacks, consequence attacks, or justification attacks. In other words, it is straightforwards to find examples that are instances of one these relations but that are not instance of either of the other two relations.

We now consider the following overarching definition of attack by default argument A on default argument B which is a generalization of support attack, claim attack, and justification attack.

Definition 11. *For default arguments A and B, the* **attacks relation** *is defined as follows.*

- A **attacks** B *iff* $\mathsf{C}(A) \cup \mathsf{S}(B) \cup \mathsf{Jp}(B) \vdash \bot$.

Given a default argument B, we consider four focused ways of attacking a default argument B as described in Table 1. We formalize these types of attack as follows, and we illustrate them in Figure 10.

Definition 12. *For default arguments A and B, the types of* **focused attack** *are defined as follows.*

- A **undermines** B *iff there exists* $\neg\beta \in \mathsf{C}(A)$ *s.t.* $\mathsf{Ep}(B) = \beta$.
- A **rebuts** B *iff there exists* $\neg\beta \in \mathsf{C}(A)$ *s.t.* $\mathsf{Ec}(B) = \beta$.
- A **undercuts** B *iff there exists* $\neg\beta \in \mathsf{C}(A)$ *s.t. there exists* $\alpha : \beta/\gamma \in \mathsf{Ip}(B)$.
- A **overcuts** B *iff there exists* $\neg\beta \in \mathsf{C}(A)$ *s.t. there exists* $\alpha : \beta/\gamma \in \mathsf{Ic}(B)$.

The undermines relation is a special case of the support attack relation, and the rebut relation is a special case of the consequence attack relation.

We also consider the following special cases of the undermines and rebuts relationships. For undermines, it is when there is a contradiction between the explicit claim of the attacker and the explicit premise of the attackee; And for rebuts, it is when there is a contradiction between the explicit claim of the attacker and the explicit claim of the attackee.

Definition 13. *For default arguments A and B, some special types of attack are defined as follows.*

- A **explicitly undermines** B *iff there exists* $\neg\beta \in \mathsf{Cn}(\mathsf{Ec}(A))$ *s.t.* $\beta \in \mathsf{Ep}(B)$.
- A **explicitly rebuts** B *iff there exists* $\neg\beta \in \mathsf{Cn}(\mathsf{Ec}(A))$ *s.t.* $\beta \in \mathsf{Ec}(B)$.
- A **implicitly undermines** B *iff A undermines B and A not explicitly undermines B.*
- A **implicitly rebuts** B *iff A rebuts B and A not explicitly rebuts B.*

Obviously, explicitly undermines (respectively explicitly rebuts) is a special case of undermines (respectively rebuts).

Example 27. *Consider* $\mathtt{A} = \langle\emptyset, \emptyset, \{\mathtt{a}\}, \{(\mathtt{a} : \mathtt{c}/\mathtt{c})\}\rangle$ *and* $\mathtt{B} = \langle\emptyset, \emptyset, \{\mathtt{b}\}, \{(\mathtt{b} : \neg\mathtt{c}/\neg\mathtt{c})\}\rangle$*.* $\mathtt{A}$ *consequence attacks* $\mathtt{B}$ *but it does not rebut or interdict. Now consider* $\mathtt{C} = \langle\emptyset, \emptyset, \{\mathtt{a} \wedge \mathtt{b}\}, \emptyset\rangle$ *and* $\mathtt{D} = \langle\{\neg\mathtt{a}\}, \{(\neg\mathtt{a} : \neg\mathtt{c}/\neg\mathtt{c})\}, \{\neg\mathtt{c}\}, \emptyset\rangle$*. So* $\mathtt{C}$ *support attacks* $\mathtt{D}$ *but it does not rebut or contradict.*

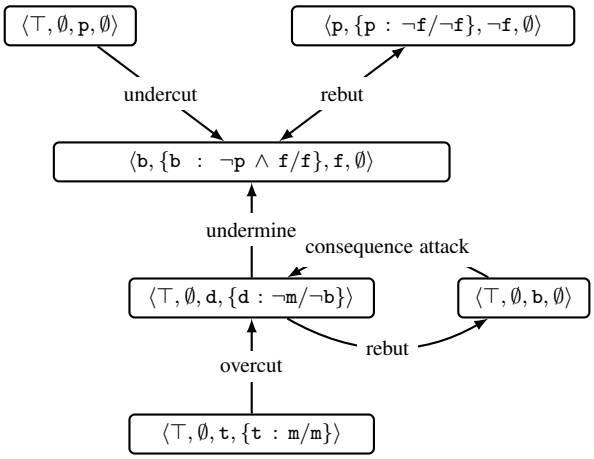


Figure 10: Instantiated argument map where b = *bird*, p = *penguin*, d = *a decoy model that looks like a bird* (i.e. a realistic model of a game bird used by hunters to lure prey to the hunter's position), t = *twitching*, m = *moving like a bird*, and f = *capable of flying*.

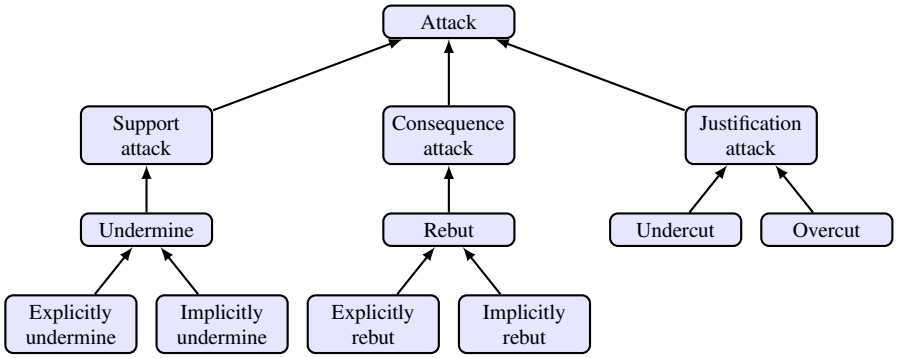


Figure 11: Relationships between different types of attack relation. The source of an arrow is a special case of the target of the arrow.

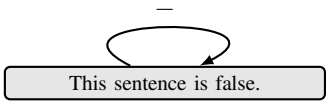
Proposition 8. *Let A be a default argument where $\mathsf{Ec}(A) \vdash \bot$. For any default argument B, if $\mathsf{C}(B) \neq \mathsf{Cn}(\{\top\})$, then A support attacks B, and if $\mathsf{S}(B) \neq \mathsf{Cn}(\{\top\})$, then A consequence attacks B. Also, if $\mathsf{Ip}(B) \neq \emptyset$, then A undercuts B, and if $\mathsf{Cn}(\mathsf{Ep}(B)) \neq \mathsf{Cn}(\{\top\})$, then A undermines B*

Proof. Assume A is a default argument where $\mathsf{Ec}(A) \vdash \bot$. If $\mathsf{C}(B) \neq \mathsf{Cn}(\{\top\})$, then there is a $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{S}(B)$, and so A support attacks B. If $\mathsf{S}(B) \neq \mathsf{Cn}(\{\top\})$, then there is a $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{C}(B)$, and so A consequence attacks B. If $\mathsf{Ip}(B) \neq \emptyset$, then there is a $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{S}(B)$, and so A undercuts B. If $\mathsf{Ep}(B) \neq \mathsf{Cn}(\{\top\})$, then there is a β s.t. $\beta \in \mathsf{Ep}(B)$ and so there is a $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{Ep}(B)$, and so A undermines B. □

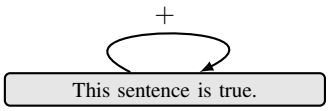
There are some correspondences between different notions of attack between default arguments as we capture in the following results. We summarise these results in Figure 11.

Proposition 9. *For A and B default arguments, the following hold: (1) If A undermines B, then A support attacks B; (2) If A rebuts B, then A consequence attacks B; And (3) if A undercuts B or A overcuts B, then A justification attacks B.*

Proof. (1) If A undermines B, then there exists $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{Ep}(B)$. So, $\mathsf{C}(A) \cup \mathsf{S}(B) \vdash \bot$, and hence A support attacks B. (2) If A rebuts B, then there exists $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{Ec}(B)$. So, $\mathsf{C}(A) \cup \mathsf{C}(B) \vdash \bot$, and hence A consequence attacks B. (3) If A undercuts B, then there exists $\neg\beta \in \mathsf{C}(A)$



(a) Argument graph with a single self-attacking arc. An example of an instantiation is $\langle \emptyset, \{\top : \mathtt{a}/\mathtt{a}\}, \{\mathtt{b}\}, \{\mathtt{b} : \neg\mathtt{a}/\neg\mathtt{a}\}\rangle$ where the atom $\mathtt{a}$ denotes "this sentence is true" and the atom $\mathtt{b}$ denotes "this sentence is false". The attack is justification attack.



(b) Argument graph with a single self-supporting arc. An example of an instantiation is $\langle \emptyset, \{\top : \mathtt{a}/\mathtt{a}\}, \{\mathtt{a}\}, \emptyset\rangle$ where the atom $\mathtt{a}$ denotes "this sentence is true". The support is justification support.

Figure 12: Examples of argument graphs with self-cycles.

s.t. there exists $\gamma : \beta/\delta \in \mathsf{Ip}(B)$, and if A overcuts B, then there exists $\neg\beta \in \mathsf{C}(A)$ s.t. there exists $\gamma : \beta/\delta \in \mathsf{Ic}(B)$; So if A undercuts or overcuts B, there is a $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{Jp}(B)$. Therefore, $\mathsf{C}(A) \cup \mathsf{Jp}(B)) \vdash \bot$. Hence, A justification attacks B. □

Proposition 10. *For default arguments A and B, suppose each rule in $\mathsf{Ic}(B)$ is normal: (1) If A undercuts B, then A support attacks B; And (2) if A overcuts B, then A consequence attacks B.*

Proof. Assume every rule in $\mathsf{Ic}(B)$ is normal. So every rule is of the form $\alpha : \beta/\beta$ in $\mathsf{Ic}(B)$. (1) Assume A undercuts B. So there exists $\neg\beta \in \mathsf{C}(A)$ s.t. there exists $\alpha : \beta/\beta \in \mathsf{Ip}(B)$. So there exists $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{S}(B)$. So A support attacks B. (2) Assume A overcuts B. So there exists $\neg\beta \in \mathsf{C}(A)$ s.t. there exists $\alpha : \beta/\beta \in \mathsf{Ic}(B)$. So there exists $\neg\beta \in \mathsf{C}(A)$ s.t. $\beta \in \mathsf{C}(B)$. So A consequence attacks B. □

The expressivity and structure of default arguments gives a wider range of attacks than considered for other proposals for structured argumentation. In particular, the notion of implicit claim rules gives us implicit claims. It also allows us to consider interesting cases of self-cycles such as in Figure 12.

An argument A can both attack and support another argument B since the claim of A can infer a premise or claim of B and also contradict a premise or claim of B.

Example 28. *For arguments $\mathtt{A} = \langle\{\mathtt{a}\}, \{\mathtt{a} : \mathtt{b}/\mathtt{c} \wedge \neg\mathtt{d}\}, \{\mathtt{c} \wedge \neg\mathtt{d}\}, \emptyset\rangle$ and $\mathtt{B} = \langle\{\mathtt{c}\}, \{\mathtt{c} : \mathtt{d} \wedge \mathtt{e}/\mathtt{d} \wedge \mathtt{e}\}, \{\mathtt{e}\}, \emptyset\rangle$. A supports B because $\mathsf{S}(A) \cap \mathsf{C}(B) \neq \emptyset$, and A attacks B because $\neg\mathtt{d} \in \mathsf{C}(\mathtt{A})$ and $\mathtt{d} \in \mathsf{C}(\mathtt{B})$.*

By using default arguments, we capture a range of ways of defining how one argument attacks another as summarized in Table 1. In particular, we capture attack on the explicit premises (undermines), on the explicit claim (rebuts), on the default derivation of the explicit claim from the premises (undercuts), and on the default derivation of the implicit claim from the explicit claim (overcuts). Furthermore, we capture attack on the inferred premises (interdict) and on the inferred claim (contradict).

6 Bridging framework

Now we can address the need to bridge argument maps and logic-based argumentation by producing instantiated argument maps. The first part of this bridging is the logical translation of premises and claims into logical formulae using a translation function (as discussed in Section 2). The second part of the briding is the instantiation by the logical argument assignment function defined below. This simply assigns a default argument to each node in the argument map.

Definition 14. *Let $M = (N, P, C, L)$ be an argument map and let $\mathcal{A}$ be a set of default arguments. A* **logical argument assignment** *for M is a function $I : N \to \mathcal{A}$.*

We give examples of logical argument assignments in Examples 29 to 32. We give further examples in Figure 14 and Figure 15.

Example 29. *Consider the argument map $M = (N, A, P, C, L)$ (in Figure 13) with $N = \{n_1, n_2, n_3\}$ where $P(n_1) = NULL$, $C(n_1) =$* Dish tastes good*; $P(n_2) = NULL$, $C(n_2) =$* Dish tastes salty*; $P(n_3) = NULL$, $C(n_3) =$* Dish tastes sweet*; such that $L(n_2, n_1) = L(n_3, n_1) = +$ and $L(n_3, n_2) = L(n_2, n_3) = -$. Moreover, suppose an atomic logical translation function Ta such that $T(P(n_1)) = \emptyset$, $T(C(n_1)) =$* `Dish_tastes_good`*, $T(P(n_2)) = \emptyset$, $T(C(n_2)) =$* `Dish_tastes_salty`*, $T(P(n_3)) = \emptyset$, $T(C(n_3)) =$* `Dish_tastes_sweet`*.*

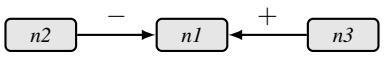
When instantiating an argument map, we would aim to instantiate the nodes so as to respect the labels on the arc. In other words, if an arc is labelled $+$, then the source argument should be a supporter of the target argument, and if an arc is labelled $-$, then the source argument should be an attacker of the target argument. For example, for Figure 13, `n2` and `n3` are supporters of `n1`, and `n2` attacks `n3` and vice versa.

We give the following definition of a specific instantiation function to illustrate how we can systematically specify how an argument map should be undertaken. In the definition we propose to instantiate defaults arguments only by relations targeting explicit premises (attacks = explicitly undermines and supports = directly supports).

Definition 15. *Let $M = (N, P, C, L)$ be an argument map, let $\mathcal{A}$ be a set of default arguments, For each $n_i \in N$, the atomic logical translation function T is such that $T(P(n_i)) = a_i$, $T(C(n_i)) = b_i$. The* **premise atomic logical argument assignment** *for M is the smallest function $I : N \to \mathcal{A}$, such that for all $n_i, n_j \in N$,*

- $\mathsf{lp}(I(n_i)) = \{(a_i : b_i / b_i)\}$*;*
- *if n_i attacks n_j (i.e. $L(n_i, n_j) = -$), then $(b_i : \neg a_j / \neg a_j) \in \mathsf{lc}(I(n_i))$;*
- *if n_j supports n_j (i.e. $L(n_i, n_j) = +$), then $(b_i : a_j / a_j) \in \mathsf{lc}(I(n_i))$.*

Example 30. *Let $M = (N, P, C, L)$ be an argument map where $N = \{n_1, n_2, n_3\}$ s.t. $L(n_2, n_1) = +$ and $L(n_3, n_1) = -$ and where the atomic logical translation function T is such that $T(P(n_1)) = \{\mathtt{a}_1\}$, $T(C(n_1)) = \{\mathtt{b}_1\}$, $T(P(n_2)) = \{\mathtt{a}_2\}$, $T(C(n_2)) = \{\mathtt{b}_2\}$, $T(P(n_3)) = \{\mathtt{a}_3\}$, $T(C(n_3)) = \{\mathtt{b}_3\}$.*



Given this argument map, a premise atomic logical argument assignment I give the following assignment to the arguments.

$$I(n_1) = \langle \{\mathtt{a}_1\}, \{(\mathtt{a}_1 : \mathtt{b}_1 / \mathtt{b}_1)\}, \{\mathtt{b}_1\}, \emptyset \rangle.$$
$$I(n_2) = \langle \{\mathtt{a}_2\}, \{(\mathtt{a}_2 : \mathtt{b}_2 / \mathtt{b}_2)\}, \{\mathtt{b}_2\}, \{(\mathtt{b}_2 : \neg\mathtt{a}_1 / \neg\mathtt{a}_1)\} \rangle.$$
$$I(n_3) = \langle \{\mathtt{a}_3\}, \{(\mathtt{a}_3 : \mathtt{b}_3 / \mathtt{b}_3)\}, \{\mathtt{b}_3\}, \{(\mathtt{b}_3 : \mathtt{a}_1 / \mathtt{a}_1)\} \rangle.$$

We have many choices for instantiation functions based on how we construct the implicit and explicit aspects of each argument. When defining a systematic way of instantiation function, we need to decide what aspects of each argument are reflect by a formula in the explicit premise or explicit claim, and what aspects are reflected by a default rule in the implicit premise or claim. Furthermore, we need to consider how arguments can be supported or attacked (i.e. the choice of definition for support or attack), and this will also have implications for how we use the justifications for the default rules.

Some general options for instantiation of default argument include: An **atomic instantiation** which assigns an argument where α and β are atoms and Φ and Ψ are sets of default rules of the form $(\gamma : \delta / \delta)$ where γ and δ are atoms (Example 29); A **propositional instantiation** which assigns an argument where α and β are propositional formula and Φ and Ψ are sets of default rules of the form $(\gamma : \delta / \delta)$ where γ and δ are propositional formula (Example 31); And a **first-order instantiation** which assigns an argument where α and β are first-order formula and Φ and Ψ are sets of default rules of the form $(\gamma : \delta / \delta)$ where γ and δ are first-order formula (Example 32).

Example 31. *We continue with the argument map in Example 30. The following is a propositional instantiation.*

$$I(n_1) = \langle \{\mathtt{c}\}, \{(\mathtt{c} : \mathtt{d} \wedge (\mathtt{e} \to \mathtt{f}) / \mathtt{e} \to \mathtt{f})\}, \{\mathtt{e} \to \mathtt{f}\}, \{(\top : \mathtt{e} / \mathtt{e})\} \rangle.$$
$$I(n_2) = \langle \{\mathtt{a} \vee \mathtt{b}\}, \{(\mathtt{a} \vee \mathtt{b} \vee \mathtt{c} : \neg\mathtt{d} / \neg\mathtt{d})\}, \{\neg\mathtt{d}\}, \emptyset \rangle.$$
$$I(n_3) = \langle \{\mathtt{g}, \mathtt{g} \to \mathtt{h}\}, \emptyset, \{\mathtt{h}, \mathtt{h} \to \mathtt{c}\}, \emptyset \rangle.$$

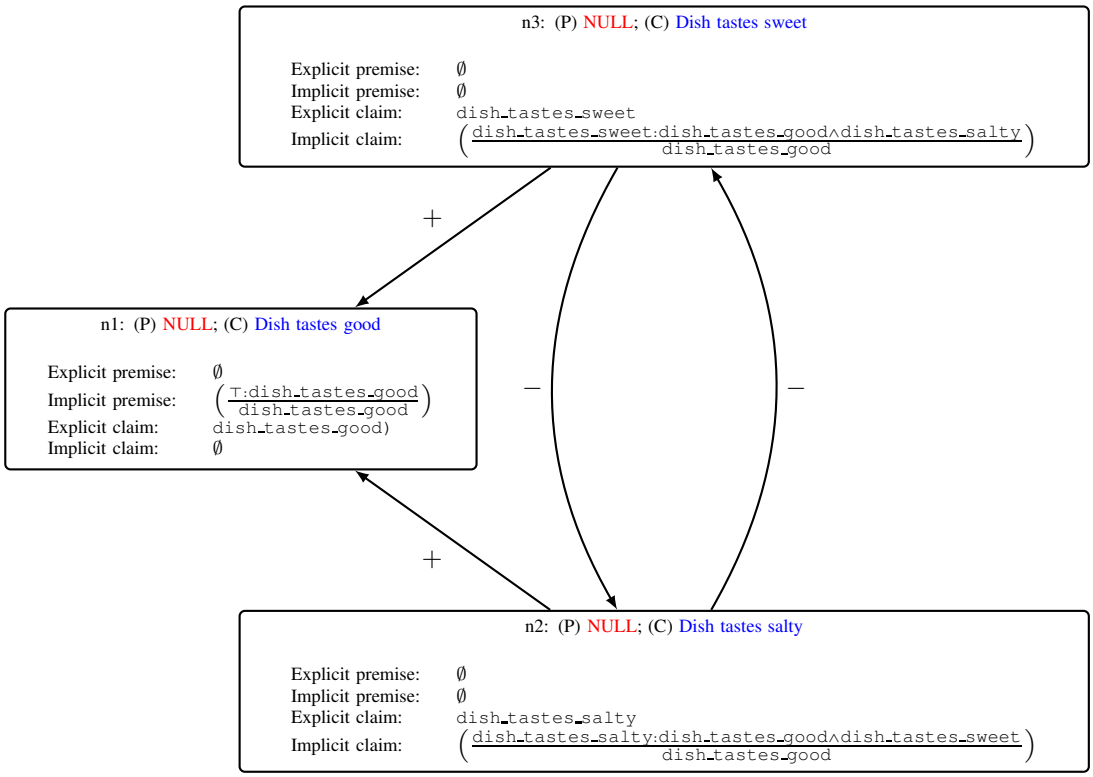


Figure 13: An example of an instantiated argument map concerning a dish tasting good.

Example 32. *We continue with the argument map in Example 30. The following is a first-order instantiation.*

$$I(n_1) = \langle\{\forall \mathtt{x}, \mathtt{y}\ \mathtt{m}(\mathtt{x},\mathtt{y})\}, \emptyset, \{\forall \mathtt{x}, \mathtt{y}\ \mathtt{m}(\mathtt{x},\mathtt{y})\}, \emptyset\rangle.$$
$$I(n_2) = \langle\{\exists \mathtt{x}, \mathtt{y}\ \neg\mathtt{m}(\mathtt{x},\mathtt{y})\}, \emptyset, \{\exists \mathtt{x}, \mathtt{y}\ \neg\mathtt{m}(\mathtt{x},\mathtt{y})\}, \emptyset\rangle.$$
$$I(n_3) = \langle\emptyset, \emptyset, \{\forall \mathtt{x}, \mathtt{y}\ \mathtt{m}(\mathtt{x},\mathtt{y})\}, \emptyset\rangle.$$

There is a wide variety of instantiation functions that we may consider. We can restrict the codomain by the choice of expressivity of language (e.g. the set of universally quantified formulae without function symbols, or the set of first-order predicate formulae composed from binary relations — and thereby draw on a knowledge graph). Furthermore, we can restrict the non-logical symbols and so restrict what atoms or predicates, constants, etc we use.

The advantage of the instantiation based on Definition 15 is that we have a relatively small number of default rules to acquire and use. If cardinality of $\mathcal{A}$ is k, then the cardinality of $\mathcal{R}$ is $2 \times k \times k$. This is because each default rule is of the form $(a_i : b_j/b_j)$ or $(a_i : \neg b_j/\neg b_j)$. So there k choices for a_i and k choices for b_j, and then for each choice of a_i and b_j, there are two default rules; the first with b_j and the second with $\neg b_j$ as the justification and consequence. This means that if we have a set of examples of cardinality for testing each default rule, we determine the probability of each default being correct in polynomial time.

To restrict instantiation, we can consider a specific knowledge-base (Π, Γ) where Π is a set of default rules and Γ is a set of classical formulae. An argument A is based on a knowledge-base (Π, Γ) iff $\mathsf{Ep}(A) \subseteq \Gamma$ and $\mathsf{Ec}(A) \subseteq \Gamma$ and $\mathsf{Ip}(A) \subseteq \Pi$ and $\mathsf{Ic}(A) \subseteq \Pi$. Let $\mathsf{Args}(\Pi, \Gamma)$ be the set of arguments based on (Π, Γ).

In general, if $|\mathsf{Args}(\Pi, \Gamma)| = k$, and $|\mathsf{Nodes}(M)| = n$, then there are n^k instantiation functions. This is because for each node there are k choices of argument to instantiate the node, and there are node nodes, hence n^k instantiations. In practice, there will be far fewer instantiation functions which satisfy constraints on arguments such as validity, mininality, coherence, and on relationships between arguments satisfying labels.

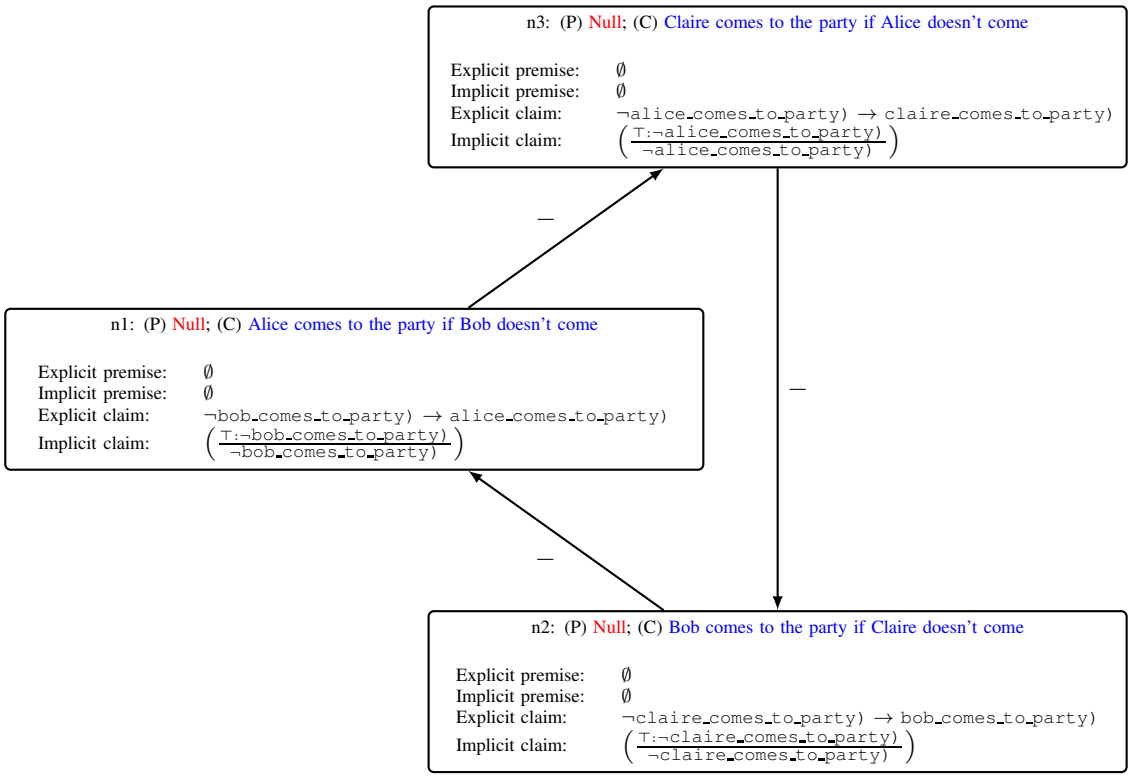


Figure 14: An example of an instantiated argument map involving a three-cycle.

A knowledge-base (Π, Γ) is **normal exhaustive** iff for every $\alpha, \beta \in \Gamma$, $(\alpha : \beta/\beta) \in \Pi$. In the case of atomic instantiations, if Π is normal exhaustive, and $|\Gamma| = x$, then $|\Pi| = (x+1)^2$ (because any atom or $\top$ can be α or β).

We now have a formal framework that specifies the input-output relationship for bridging argument maps and logic-based argumentation: The input is the argument map (from argument mining), and the output is the instantiated argument map. Furthermore, our new framework provides a logical representation of both the implicit and explicit aspects of the premises and claims. This can then be used to investigate ambiguity arising from implicitness which results in choices for an instantiation of an argument map. By analysing the range and diversity of possible instantiations for an argument map, we may be able to measure the rebustness of any part of an instantiation. We may also consider constraints on instantiations that capture desirable behavour (such as been proposed for instantiating bipolar argument graphs with deductive arguments [Hun23]. The framework may also be viewed as a starting point for deciding what is an semantically appropriate instantiation by using some kind of semantic analysis of the explicit premises, explicit claims and the relations.

7 Literature review

In this section, we consider our proposal for instantiated argument maps with developments in abstract bipolar argumentation, structured argumentation including approaches using default logic, and formalisms for handling of enthymemes.

7.1 Bipolar argumentation

Bipolar argumentation is a generalization of abstract argumentation that incorporates a support relation in addition to the attack relation [CL05a, CL05b, ACLL08, CL13]. A bipolar argument graph is a directed

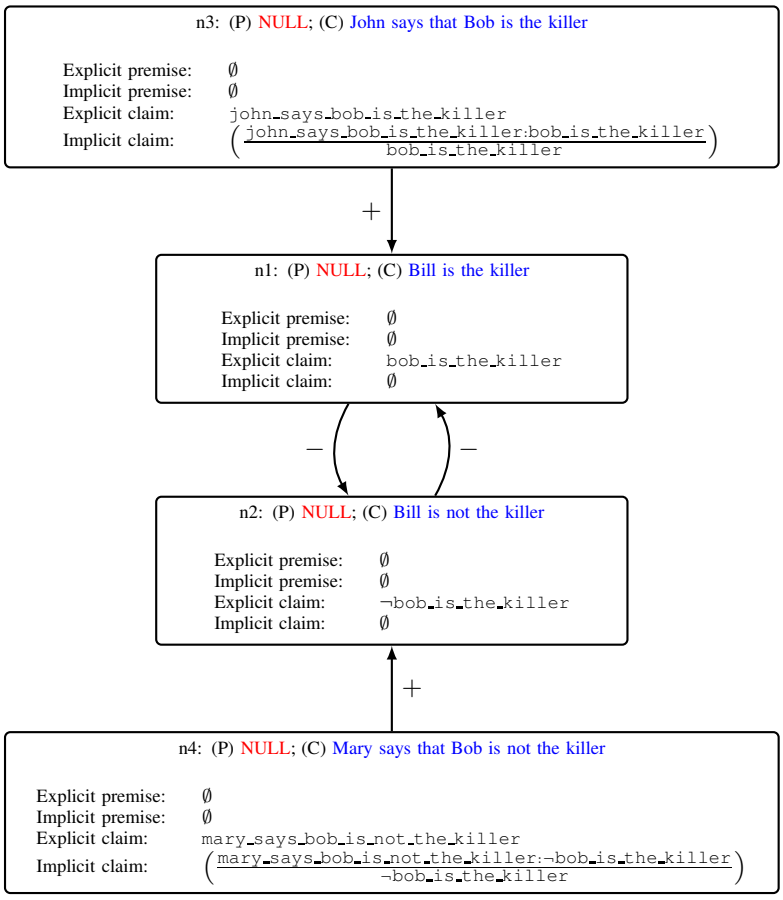


Figure 15: An example of an instantiated argument map adapted from [Pra23].

graph where each node denotes an argument, and each arc denotes the influence of one argument on another. The label denotes the type of influence with options including positive (supporting) and negative (attacking). In order to determine the acceptable arguments from a biploar argument graph, dialectical semantics can be generalized to handle both support and attack relations [CL13]. Alternatively, the approach of abstract dialectical frameworks [BW10, BW14], gradual semantics [AB13, ABDV17, AD21, BDKM16, CL05b, LM11, RTAB16, dCPTV11, BRT$^+$15, Pot18, PLZL14, PLZL15, BRT19], or epistemic graphs [HPT20] can be used.

An issue with bipolar argument graphs is that each node is an abstract argument, and so the meaning of the argument is unspecified. To address this issue, we have investigated how nodes can be instantiated with default arguments. By doing this, we can systematically investigate the nature of the contents (premises and claim) and their interplay with the structure of the graph. But this then raises questions about what kinds of instantiations are appropriate and what they mean. It also raises questions about how we can compare instantiated bipolar argument graphs to show, for instance, that two instantiations are equivalent, and how we can investigate the interplay of the premises of an argument and the claims of the arguments that support or attack it.

7.2 Structured argumentation

Our proposal goes beyond existing proposals for structured argumentation. In Section 4, we have shown how we can capture classical logic deductive arguments as default logic. It is then straightforward to capture the range of attack relationships [BH01, BH08a, GH11] and support relationships [Hun23] in

our new framework. Whilst default knowledge can be captured in the deductive argumentation approach using normality atoms [BH14], our new proposal provides a more systematic way of representing these as justifications.

The two main approaches to structured argumentation are assumption-based argumentation (ABA) [Ton14], and ASPIC+ [MP14]. In ABA, assumptions are atoms that can be used in the condition of proof rules in a way that replicates justifications in default rules. So a proof rule can be attacked when the contrary of an assumption holds. Furthermore, assumption-based argumentation has been shown to capture default lotic [BDKT97]. In other words, any default theory can be equivalently represented by an ABA knowledge-base. Similarly to ASPIC+, proof rules are labelled, and these labels can be used at atoms in the formulae of the object language. Furthermore, an argument using the proof rule is attacked when the contrary of its label holds. In contrast to ASPIC+ and ABA, our proposal in this paper clearly demarks the implicit premises and implicit claims in a strutured argument representation of an enthymeme, and it further goes beyond ASPIC+ and ABA by providing a framework of definitions for support and attack relations between default arguments.

There are other proposals in the computational argumentation literature for using default logic such as [Pra93, SM08, Hun18]. But our proposal goes beyond these in that we provide a comprehensive coverage of differnt types of argument and of different types of support and attack relationships. In the proposal by Prakken [Pra93], a default argument is based on a ground default theory with a unique extension where the premises are the ground default theory and the claim is an element in the extension. An argument A is a counterargument to an argument B if the claim of A contradicts B. These notions of argument and counterargument are special cases in our more general framework. In the proposal by Santos *et al.* [SM08] for using default logic in argumentation, the premises of an argument is a minimal default theory (D, W), with a unique and consistent extension, where the claim is in the extension of the default theory. They also introduce the notion of a justificative argument which is an argument based on a default theory (D, W) where the justifications of each default in D is implied by W. A justificative argument can be viewed as an argument that does not rely on "unknown" information (i.e. the argument can be written into deductive argumentation where the justification can be equally represented as a precondition). Furthermore, for an argument with premises (D, W), an undercut is defined as an argument with a claim that negates a some of the formulae in W and and/or some of the justifications in the rules in D. The proposals by Prakken and by Santos *et al.* focus on specific kinds of arguments, and do not consider the more general notion of arguments that we present in this paper. Furthermore, they do not give consideration to handling implicit premises and implicit claims, and so do not consider issues of modelling enthymemes.

7.3 Handling enthymemes

Most proposals for handling enthymemes in the compuptational argumentation literature either focus on the coding/decoding of enthymemes via abduction [Hun07, BH12, HMR14], and how this can be undertaken within a dialogue [BH08b, dS11b, dS11a, XHMB20, PMB22, LGG23]. For instance, dialogues can be shown to shorter when using enthymemes [PMB22]. However, these proposals do not provide a systematic framework for translating argument maps into logic, and concomitantly, addressing the problem of identifying the missing premises and/or claim, and discerning the relationships between them.

Another important aspect of dealing with enthymemes is the uncertainty that arises from decoding. When an audience is listening to participants in a discussion or debate, the participants present arguments including enthymemes. The way these are presented gives some idea to the audience of which are meant to attack which. However, if an argument being attacked is an enthymeme, and the attacking and attacked arguments are from different participants, then there is uncertainty about whether it is indeed a valid attack. Each enthymeme is a representative for an intended argument, but for the audience it may be uncertain which decoding is the intended argument. The audience may have zero or more choices. This means that the audience takes an argument graph as input, and tries to determine the intended argument graph (i.e. the graph obtained by instantiating each node with its intended argument and deleting the arcs that are not valid attacks). This intended argument graph has a structure that would be isomorphic to a spanning subgraph of the original argument graph. One way to model this uncertainty is to use the constellations approach to probabilistic argumentation, where a probability distribution over these graphs is used to reflect the uncertainty [Hun13]. However, this assumes that the graph contains abstract arguments. Another approach

that is also based on abstract argumentation. is to use the approach of incomplete graphs [Mai16].

A key problem with these proposals for handling enthymemes in the computational argumentation literature (such as [Hun07, BH12, HMR14, XHMB20, PMB22]) is that they assume an extensive set of common or commonsense knowledge with a preference ordering over the appropriateness of any formula for a specific audience. These are demanding assumptions. It is difficult to acquire commonsense knowledge, and in particular, there are often gaps in the explicit knowledge that connects concepts. Relationships between concepts are often known but they are implicit. For instance, different symbols for the same concept are used (e.g. `dad` and `father`) or for similar concepts (`showery` and `rainy`), though sometimes such correspondences are context dependent (e.g. in the context of a church, `dad` and `father` often refer to different concepts).

To address this need for a scalable and robust way of connecting concepts, distance measures between semantic concepts can be used [Hun22]. We assume that each semantic concept is represented by a word, and so in propositional logic, each propositional atom is a word (or compound word), and in predicate logic, each predicate, function, and constant, symbol is a word (or compound word). So with a distance measure, we want the distance between `dad` and `father` to be low, and between `dad` and `grass` to be high. Fortunately, there are now some robust and scalable options for distance measures that we can use based on word embeddings and sentence embeddings. None of these will be totally correct, but the error rate could be tolerable, and the benefits of using them far outweighing this.

7.4 Translating text into logic

Within the NLP community, there has always been interest in technology for translating free text into formal logic. In the past, use of approaches such as phrase-structure grammars, allows for translation of fragments of natural language into logic, but it is only with the advent of deep leanring and large language models that it appears more general, robust, and scalable, methods can be developed for translating natural language into logic. Preliminary investigations using deep learning include [SAK20, LL21] and using LLMs include [LLG$^+$22, PAWW23, OGL$^+$23, LCH$^+$24].

Another approach is to use abstract meaning representation (AMR) as a semantic representation of a sentence as a directed graph. The aim of AMR is that the graph reflects the content of a sentence in terms of the actors, and their roles, in the sentence, rather than a complete coverage of the grammar of the sentence [LK98]. So it is intended that sentences that are grammatically different but the same or similar semantics are represented by the same AMR graph. By drawing on LLMs, some AMR parsers offer a general, robust, and scalable, approach to generating AMR graphs from a wide variety of text (e.g. [LAF$^+$23]). Once, an AMR graph has been obtained from free text, a logical formula can be obtained which can be reasoned with using neurosymbolic approach [CH23, FH24].

Finally, a recent proposal for analysing enthymemes draws on the semi-formal approach of argument schema [RTLR24]. First, a dataset has been synthetically generated using large language models (GPT3.5 and GPT 4) where the model is prompted to fill out an argument scheme from a selection of 20 schema. Then, fine-tuned Roberta models were developed for classification of free text arguments according the type of argument scheme. Whilst, this proposal does not produce logical arguments, it could be harnessed as part of process of producing logical arguments. This dataset has also been used in developing a reinforcement learning approach for classifying free text arguments according the type of argument scheme using Mistral [TEB$^+$24].

8 Discussion

In the real-world, arguments are often enthymemes (i.e. arguments with some premises being implicit). The missing premises have to be abduced by the recipient. These may be obtained from contextual and background knowledge, and more challengingly, common-sense knowledge. We need to understand enthymemes if we want to make sense of them or respond to them. If we don't understand an enthymeme properly, we can have misunderstandings, and we can talk at cross-purposes.

Argument mining find sentences representing premises and claims. But there is a lack of a framework to then identify the underlying logical arguments. We have addressed this shortcoming by providing the

following novel contributions in this paper: (1) A form of structured argumentation based on default logic; (2) A framework for representing argument maps and representing logical instantiation of them using structured argumentation based ond default logic; and (3) Investigation of specific kinds of instantiation based on types of argument, attack and support, and instantiation method.

Following from our proposal, there are a number of areas of future work. First, we would like to investigate methods for translation of premises and claims in logic. A simple option is to use a large language model to classify each type of premise or claim, and thereby obtain an atomic translation. A more complex option is to use an AMR parser to translate text into an AMR graph which in turn can be translated into a propositional or first order formula [CH23, FH24].

Second, we would like to investigate how to handle the uncertainty arising in decoding enthymemes. This includes the uncertainty from translating the premises and claims into logic and the uncertainty in finding the instantiation. For instance, if we use a text classifier, then we can have a probability distribution over the classifications, and this could be reflected by a distribution over the instantiations. There is also the related question of finding instantiations that satisfying the labels (an attacker for a negative label and a supporter for a positive label). This may then lead to a constraint satisfaction problem for satisfying the maximum number of labels.

Third, we would like to investigate how we can acquire and represent common and common-sense knowledge in the form of default logic. It would be desirable to render the approach scalable which therefore calls for automated means for acquiring this kind of knowledge.

References

[AB13] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In *Proceedings of SUM'13*, volume 8078 of *LNCS*, pages 134–147. Springer, 2013.

[ABDV17] Leila Amgoud, Jonathan Ben-Naim, Dragan Doder, and Srdjan Vesic. Acceptability semantics for weighted argumentation frameworks. In *Proceedings of IJCAI'17*, pages 56–62. IJCAI, 2017.

[ABG+17] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Ricardo Simari, Matthias Thimm, and Serena Villata. Towards artificial argumentation. *AI Magazine*, 38(3):25–36, 2017.

[ACLL08] Leila Amgoud, Claudette Cayrol, Marie-Christine Lagasquie-Schiex, and P. Livet. On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, 23(10):1062–1093, 2008.

[AD18] Leila Amgoud and Victor David. Measuring similarity between logical arguments. In *Proceedings of KR'18*, 2018.

[AD21] Leila Amgoud and Victor David. A general setting for gradual semantics dealing with similarity. In *Proceedings of AAAI'21*, volume 35, pages 6185–6192, 2021.

[BDK97] Gerhard Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview*, volume 73 of *CSLI Lecture Notes*. CSLI Publications, Stanford, CA, 1997.

[BDKM16] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A comparative study of ranking-based semantics for abstract argumentation. In *Proceedings of AAAI'16*, pages 914–920. AAAI Press, 2016.

[BDKT97] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[BEK+21] Roy Bar-Haim, Lilach Eden, Yoav Kantor, Roni Friedman, and Noam Slonim. Every bite is an experience: Key point analysis of business reviews. In *Proceedings of ACL'21*, pages 3376–3386. Association for Computational Linguistics, 2021.

[Bes89] Philippe Besnard. *An Introduction to Default Logic*. Springer, 1989.

[BH01] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128(1-2):203–235, 2001.

[BH05] Philippe Besnard and Anthony Hunter. Practical first-order argumentation. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, pages 590–595. MIT Press, 2005.

[BH08a] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[BH08b] Elizabeth Black and Anthony Hunter. Using enthymemes in an inquiry dialogue system. In *Proceedings of AAMAS'08*, pages 437–444. IFAAMAS, 2008.

[BH12] Elizabeth Black and Anthony Hunter. A relevance-theoretic framework for constructing and deconstructing enthymemes. *Journal of Logic and Computation.*, 22(1):55–78, 2012.

[BH14] Philippe Besnard and Anthony Hunter. Constructing argument graphs with deductive arguments. *Argument and Computation*, 5(1):5–30, 2014.

[BQQ83] Philippe Besnard, Rene Quiniou, and Patrice Quinton. A theorem-prover for a decidable subset of default logic. In *Proceedings of AAAI'83*, pages 27–30. AAAI Press, 1983.

[Bre91] Gerhard Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, 1991.

[BRT+15] Pietro Baroni, Marco Romano, Francesca Toni, Marco Aurisicchio, and Giorgio Bertanza. Automatic evaluation of design alternatives with quantitative argumentation. *Argument and Computation*, 6(1):24–49, 2015.

[BRT19] Pietro Baroni, Antonio Rago, and Francesca Toni. From fine-grained properties to broad principles for gradual argumentation: A principled spectrum. *International Journal of Approximate Reasoning*, 105:252–286, 2019.

[BW10] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proceedings of KR'10*. AAAI Press, 2010.

[BW14] Gerhard Brewka and Stefan Woltran. GRAPPA: A semantical framework for graph-based argument processing. In *Proceedings of ECAI'14*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 153–158. IOS Press, 2014.

[CH20] Lisa A. Chalaguine and Anthony Hunter. A persuasive chatbot using a crowd-sourced argument graph and concerns. In *Proceedings of COMMA'20*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 9–20. IOS Press, 2020.

[CH23] David Chanin and Anthony Hunter. Neuro-symbolic commonsense social reasoning. *CoRR*, abs/2303.08264, 2023.

[CL05a] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Gradual valuation for bipolar argumentation frameworks. In *Proceedings of ECSQARU'05*, volume 3571 of *LNCS*, pages 366–377, 2005.

[CL05b] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Graduality in argumentation. *Journal of Artificial Intelligence Research*, 23:245–297, 2005.

[CL13] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: Towards a better understanding. *International Journal of Approximate Reasoning*, 54(7):876–899, 2013.

[CLS20] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Logical encoding of argumentation frameworks with higher-order attacks and evidential supports. *International Journal on Artificial Intelligence Tools*, 29(03n04):2060003, 2020.

[CWZZ10] Yin Chen, Hai Wan, Yan Zhang, and Yi Zhou. dl2asp: Implementing default logic via answer set programming. In *Logics in Artificial Intelligence*, pages 104–116. Springer, 2010.

[Dav17] Ernest Davis. Logical formalizations of commonsense reasoning: A survey. *Journal of Artificial Intelligence Research*, 59:651–723, 2017.

[dCPTV11] Célia da Costa Pereira, Andrea Tettamanzi, and Serena Villata. Changing one's mind: Erase or rewind? Possibilistic belief revision with fuzzy argumentation based on trust. In *Proceedings of IJCAI'11*, pages 164–171. AAAI Press, 2011.

[DCS17] Marco Damonte, Shay B. Cohen, and Giorgio Satta. An incremental parser for abstract meaning representation. In *Proceedings of EACL'17*, 2017.

[dS11a] Florence Dupin de Saint-Cyr. A first attempt to allow enthymemes in persuasion dialogs. In Franck Morvan, A Min Tjoa, and Roland R. Wagner, editors, *Proceedings of DEXA'11*, pages 332–336. IEEE Computer Society, 2011.

[dS11b] Florence Dupin de Saint-Cyr. Handling enthymemes in time-limited persuasion dialogs. In *Proceedings of SUM'11*, volume 6929 of *Lecture Notes in Computer Science*, pages 149–162. Springer, 2011.

[DZF+22] Andrew Drozdov, Jiawei Zhou, Radu Florian, Andrew McCallum, Tahira Naseem, Yoon Kim, and Ramón Fernandez Astudillo. Inducing and using alignments for transition-based AMR parsing. In *Proceedings of NAACL'22*, pages 1086–1098. Association for Computational Linguistics, 2022.

[FH24] Xuyao Feng and Anthony Hunter. Identification of entailment and contradiction relations between natural language sentences: A neurosymbolic approach. *CoRR*, arXiv:2405.01259, 2024.

[FM94] Christine Froidevaux and Jérôme Mengin. Default logics: A unified view. *Computational Intelligence*, 10:331–369, 1994.

[GH11] Nicos Gorogiannis and Anthony Hunter. Instantiating abstract argumentation with classical logic arguments: Postulates and properties. *Artificial Intelligence*, 175(9-10):1479–1497, 2011.

[GS04] Alejandro Javier García and Guillermo Ricardo Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4:95–138, 2004.

[GTF+23] Shai Gretz, Assaf Toledo, Roni Friedman, Dan Lahav, Rose Weeks, Naor Bar-Zeev, João Sedoc, Pooja Sangha, Yoav Katz, and Noam Slonim. Benchmark data and evaluation framework for intent discovery around COVID-19 vaccine hesitancy. In *Findings of EACL'23*, pages 1358–1370. Association for Computational Linguistics, 2023.

[HMR14] Seyed Ali Hosseini, Sanjay Modgil, and Odinaldo Rodrigues. Enthymeme construction in dialogues using shared knowledge. In *Proc. of COMMA'14*, volume 266 of *FAIA*, pages 325–332. IOS Press, 2014.

[HPT20] Anthony Hunter, Sylwia Polberg, and Matthias Thimm. Epistemic graphs for representing and reasoning with positive and negative influences of arguments. *Artificial Intelligence*, 281:103236, 2020.

[Hun07] Anthony Hunter. Real arguments are approximate arguments. In *Proceedings of AAAI'07*, pages 66–71. MIT Press, 2007.

[Hun13] Anthony Hunter. A probabilistic approach to modelling uncertain logical arguments. *International Journal of Approximate Reasoning*, 54(1):47–81, 2013.

[Hun18] Anthony Hunter. Non-monotonic reasoning in deductive argumentation. *CoRR*, abs/1809.00858, 2018.

[Hun22] Anthony Hunter. Understanding enthymemes in deductive argumentation using semantic distance measures. In *Proceedings of AAAI'22*, pages 5729–5736. AAAI Press, 2022.

[Hun23] Anthony Hunter. Some options for instantiation of bipolar argument graphs with deductive arguments. *CoRR*, abs/2308.04372, 2023.

[LAF+23] Young-Suk Lee, Ramón Fernandez Astudillo, Radu Florian, Tahira Naseem, and Salim Roukos. AMR parsing with instruction fine-tuned pre-trained language models. *CoRR*, abs/2304.12272, 2023.

[LCH+24] Abhinav Lalwani, Lovish Chopra, Christopher Hahn, Caroline Trippel, Zhijing Jin, and Mrinmaya Sachan. NL2FOL: translating natural language to first-order logic for logical fallacy detection. *CoRR*, abs/2405.02318, 2024.

[LGG23] Diego S. Orbe Leiva, Sebastian Gottifredi, and Alejandro Javier García. Automatic knowledge generation for a persuasion dialogue system with enthymemes. *International Journal of Approximate Reasoning*, 160:108963, 2023.

[LK98] Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics.

[LL21] Oleksii Levkovskyi and Wei Li. Generating predicate logic expressions from natural language. In *SoutheastCon 2021*, pages 1–8, 2021.

[LLG+22] Xuantao Lu, Jingping Liu, Zhouhong Gu, Hanwen Tong, Chenhao Xie, Junyang Huang, Yanghua Xiao, and Wenguang Wang. Parsing natural language into propositional and first-order logic with dual reinforcement learning. In *Proceedings of COLING'22*, pages 5419–5431. International Committee on Computational Linguistics, 2022.

[LM11] João Leite and João G. Martins. Social abstract argumentation. In *Proceedings of IJCAI'11*, pages 2287–2292. AAAI Press, 2011.

[LR19] John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, December 2019.

[Mai16] Jean-Guy Mailly. Using enthymemes to fill the gap between logical argumentation and revision of abstract argumentation frameworks. *CoRR*, abs/1603.08789, 2016.

[MP14] Sanjay Modgil and Henry Prakken. The *ASPIC*$^+$ framework for structured argumentation: a tutorial. *Argument and Computation*, 5(1):31–62, 2014.

[OGL+23] Theo Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of EMNLP'23*, pages 5153–5176. Association for Computational Linguistics, 2023.

[PAWW23] Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of EMNLP'23*, pages 3806–3824. Association for Computational Linguistics, 2023.

[PLZL14] Fuan Pu, Jian Luo, Yulai Zhang, and Guiming Luo. Argument ranking with categoriser function. In *Proceedings of KSEM'14*, volume 8793 of *LNCS*, pages 290–301. Springer, 2014.

[PLZL15] Fuan Pu, Jian Luo, Yulai Zhang, and Guiming Luo. Attacker and defender counting approach for abstract argumentation. In *Proceedings of CogSci'15*, page 1, 2015.

[PMB22] Alison R. Panisson, Peter McBurney, and Rafael H. Bordini. Towards an enthymeme-based communication framework in multi-agent systems. In *Proceedings of KR'22*, 2022.

[Pot18] Nico Potyka. Continuous dynamical systems for weighted bipolar argumentation. In *Proceedings of KR'18*, pages 148–157. AAAI Press, 2018.

[Pra93] Henry Prakken. An argumentation framework in default logic. *Annals of Mathematics and Artificial Intelligence*, 9:93–132, 1993.

[Pra23] Henry Prakken. When is argumentation deductive? *Journal of Applied Non-Classical Logics*, 33(3-4):212–223, 2023.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[RS94] Vincent Risch and Camilla Schwind. Tableaux-based characterization and theorem proving for default logic. *Journal of Automated Reasoning*, 13(2):223–242, 1994.

[RTAB16] Antonio Rago, Francesca Toni, Marco Aurisicchio, and Pietro Baroni. Discontinuity-free decision support with quantitative argumentation debates. In *Proceedings of KR'16*, pages 63–73. AAAI Press, 2016.

[RTLR24] Ramon Ruiz-Dolz, Joaquín Taverner, John Lawrence, and Chris Reed. NLAS-multi: A multilingual corpus of automatically generated natural language argumentation schemes. *CoRR*, abs/2402.14458, 2024.

[SAK20] Hrituraj Singh, Milan Aggarwal, and Balaji Krishnamurthy. Exploring neural models for parsing natural language into first-order logic. *CoRR*, abs/2002.06544, 2020.

[SM08] Emanuel Santos and João Pavão Martins. A default logic based framework for argumentation. In *Proceedings of ECAI'08*, pages 859–860, 2008.

[TDJ22] Tanel Tammet, Dirk Draheim, and Priit Järv. Gk: Implementing full first order default logic for commonsense reasoning (system description). In *Automated Reasoning*, pages 300–309. Springer, 2022.

[TEB+24] Guilherme Trajano, Débora Englemann, Rafel Bordini, Stefan Sarkadi, Jack Mumford, , and Alison Panisson. Translating natural language arguments to computational arguments using LLMs. In *Proceedings of COMMA'24*. IOS Press, 2024.

[Ton14] Francesca Toni. A tutorial on assumption-based argumentation. *Argument and Computation*, 5(1):89–117, 2014.

[Wal01] Douglas Walton. Enthymemes, common knowledge and plausible inference. *Philosophy and Rhetoric*, 34(2):93–112, 2001.

[WvEH16] Adam Z. Wyner, Tom M. van Engers, and Anthony Hunter. Working on the argument pipeline: Through flow issues between natural language argument, instantiated arguments, and argumentation frameworks. *Argument and Computation*, 7(1):69–89, 2016.

[XHMB20] Andreas Xydis, Christopher Hampson, Sanjay Modgil, and Elizabeth Black. Enthymemes in dialogues. In *Proc. of COMMA'20*, volume 326 of *FAIA*, pages 395–402. IOS Press, 2020.